AD-A085 507 ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB F/G 17/2
A NUMERICAL STUDY OF TWO MEASURES OF DELAY FOR NETWORK ROUTING.(U)
SEP 79 K S VASTOLA DAAG29-78-C-0016
UNCLASSIFIED R-859 NL

1 OF 1

AD A085507

END
DATE FILMED
7-80
DTIC

REPORT R-859 AUGUST, 1979 UILU-ENG 78-2252

CSL COORDINATED SCIENCE LABORATORY

LEVEL II

12

ADA085507

A NUMERICAL STUDY OF TWO MEASURES OF DELAY FOR NETWORK ROUTING

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A085507
3. RECIPIENT'S CATALOG NUMBER	[illegible]
4. TITLE (and Subtitle)	A NUMERICAL STUDY OF TWO MEASURES OF ~~DISPLAY~~ Delay FOR NETWORK ROUTING,
5. TYPE OF REPORT & PERIOD COVERED	Technical Report
6. PERFORMING ORG. REPORT NUMBER	R-859, UILU-ENG-78-2252
7. AUTHOR(s)	KENNETH STEVEN VASTOLA
8. CONTRACT OR GRANT NUMBER(s)	DAAG-29-78-C-0016, N00014-79-C-0424, NSF ENG 77-15947
9. PERFORMING ORGANIZATION NAME AND ADDRESS	
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	[illegible]
11. CONTROLLING OFFICE NAME AND ADDRESS	Joint Services Electronics Program
12. REPORT DATE	Sep[redacted]79
13. NUMBER OF PAGES	57
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	
15. SECURITY CLASS. (of this report)	UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Multicommodity flow problems
Computer-communication network delay

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This work presents computational results relating to the solution of convex multicommodity network flow problems by using an algorithm developed by D. P. Bertsekas [1] from the ideas of Gallagher's method for distributed optimization of delay in data communication networks [2] and gradient projection ideas from nonlinear programming [3],[4]. This algorithm is described and results are given for two measures of "delay" in the network. In the first case, the formula for delay was based on the expression for queuing delay in M/M/1 queues suggested by Kleinrock [15]. Here several versions of the algorithm

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 047700
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20. Abstract (continued)

(including with and without line search) are discussed. In the second case, the algorithm is applied to the dual problem in an approximation scheme based on a method of multipliers with exponential penalty function due to Kort and Bertsekas [13]. Here the measure of delay being minimized is the largest link utilization in the network. Finally the two measures of delay are compared. In the examples we find that a fairly good solution to each of the problem may be found near the optimal point for the other problem.

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

A NUMERICAL STUDY OF TWO MEASURES OF DISPLAY FOR NETWORK ROUTING

by

Kenneth Steven Vastola

This work was supported in part by the Joint Services Electronics Program (U. S. Army, U. S. Navy, and U. S. Air Force) under Contracts DAAG-29-78-C-0016 and N00014-79-C-0242, and in part by the National Science Foundation under Grant ENG 77-15947.

A NUMERICAL STUDY OF TWO MEASURES OF DELAY FOR NETWORK ROUTING

BY

KENNETH STEVEN VASTOLA

B.A., Rutgers University, 1976

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1979

Thesis Advisor: Dimitri P. Bertsekas

Urbana, Illinois

ACKNOWLEDGMENT

The author wishes to express his appreciation to his thesis advisor, Professor Dimitri P. Bertsekas, for his support and advice. In addition, he thanks Ms. Rose Harris, Mrs. Trudy Little and Mrs. Wilma McNeal for typing this thesis.

Finally, he would like to thank his wife, Deborah, and his parents and family for their encouragement and support.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

In a packet-switched computer-communication network the "messages" to be sent are broken into blocks called packets. For our purposes we will consider such a packet as the basic unit of input to and flow in the network. Our objective is to find a flow which minimizes the "delay" in a particular network for a particular set of inputs.

The basic algorithm (developed by D. P. Bertsekas [1] and R. Gallager [2]) is presented using a general measure of the delay (i.e. a general objective function) in Chapter 2 along with a description of certain procedures developed in order to program the algorithm efficiently. Also, in Section 2 of Chapter 2, possible line searches are discussed.

In Chapter 3 results are given and discussed for a delay function which is an average over all the links in the network. The delay on each link is computed via a formula suggested by Kleinrock [15].

In Chapter 4, the algorithm is adapted to solve the "dual" problem in an approximation scheme based on a method of multipliers with exponential penalty function due to Kort and Bertsekas [13]. In this case we attempt to minimize the largest link utilization in the network where the utilization of a particular link is the ratio of the flow to the capacity of that link.

CHAPTER 2

THE ALGORITHM AND THE PROGRAM

2.1. Mathematical Description of the Basic Algorithm

We consider a network consisting of N nodes represented by $\mathcal{N} = \{1,2,\ldots,N\}$ and a set $\mathcal{L}$ of L directed links. We denote a link from node i to node j by the ordered pair (i,j). It is assumed that given nodes i and j there exist nodes $k_1,\ldots,k_m$ such that $(i,k_1),(k_1,k_2),\ldots,(k_{m-1},k_m),(k_m,j)$ are elements of $\mathcal{L}$.

Let $f_{ik}(j)$ represent the portion of the flow in link (i,k) which is destined for node j. Consider the following multicommodity flow problem in the variables $f_{ik}(j)$

$$\text{minimize} \sum_{(i,k)\in\mathcal{L}} D_{ik}\Big[\sum_{j=1}^{N} f_{ik}(j)\Big] \qquad \text{(MFP)}$$

$$\text{subject to} \sum_{k\in\mathcal{O}(i)} f_{i\ell}(j) - \sum_{m\in I(i)} f_{mi}(j) = r_i(j) \quad \forall i\neq j,\ i\in\mathcal{N},\ j\in\mathcal{N}$$

$$f_{ik}(j) \geq 0 \quad \forall (i,k)\in\mathcal{L},\ j\in\mathcal{N}$$

$$f_{jk}(j) = 0 \quad \forall (j,k)\in\mathcal{L}$$

where $\mathcal{O}(i) = \{k\in\mathcal{N} \mid (i,k)\in\mathcal{L}\}$, $I(i) = \{m\in\mathcal{N} \mid (m,i)\in\mathcal{L}\}$, and $r_i(j)$ represents the known traffic input at node i which is destined for node j. We assume $r_i(j) \geq 0$ $\forall i\in\mathcal{N}$, $j\in\mathcal{N}$ and that $\forall (i,\ell)\in\mathcal{L}$ $D_{i\ell}$ is a twice continuously differentiable real-valued function on $[0,\infty)$ whose first and second derivatives are positive.

2.1.1. The Goldstein-Levitin-Polyak Gradient Projection Method

The Goldstein-Levitin-Polyak gradient projection method may be used to solve the constrained optimization problem:

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } Ax = b,\ x \geq 0 \end{aligned} \tag{1}$$

where $f : R^n \to R$ is a twice continuously differentiable function, A is an $m \times n$ matrix, and b an m-dimensional column vector.

Beginning with an initial feasible vector $x_o \in R^n$ we generate x_{k+1} from x_k by letting

$$x_{k+1} = \overline{x}_k \text{ where } \overline{x}_k \text{ solves:} \tag{2}$$

$$\begin{aligned} &\text{minimize } \nabla f(x)^T(x-x_k) + 1/2(x-x_k)^T M_k (x-x_k) \\ &\text{subject to } Ax = b,\ x \geq 0, \end{aligned} \tag{3}$$

where M_k is a symmetric matrix which is positive definite on the null space of A, i.e. $x^T M_k x > 0$, $\forall x \neq 0$ such that $Ax = 0$. The objective function in the problem (3) is to be thought of as a quadratic approximation of f on the constraint set. In particular if we let $M_k = \nabla^2 f(x_k)$ (assuming positive definiteness on the null space of A) the problem (2),(3) becomes a constrained version of Newton's method. In this case when problem (1) has a unique solution $\overline{x}$, this method converges to $\overline{x}$ at a superlinear rate if x_o is close enough to $\overline{x}$ (see [3], Theorem 7.1). However, in most situations (3) is too difficult to solve with $M_k = \nabla^2 f(x_k)$ and M_k must be merely an approximation to $\nabla^2 f(x_k)$ (e.g. M_k equal to $\nabla^2 f(x_k)$ on the diagonal and zero off).

More information on this method may be found in [1], [3], and [4]. In particular, in [1] it is shown that under mild assumptions, choosing M_k sufficiently large, this method converges.

2.1.2. The Basic Algorithm

For simplicity of notation and explanation we consider the special case where there is only one destination, i.e. $r_i(j) = 0 \quad \forall i,j \in \mathcal{N}$ with $j \neq N$. We may now write the original problem in the following form

$$\text{minimize} \sum_{\substack{(i,\ell)\in\mathcal{L} \\ i\neq N}} D_{i\ell}(f_{i\ell}) \qquad \text{(SFP)}$$

$$\text{subject to} \begin{cases} \sum\limits_{\ell\in\mathcal{O}i} f_{i} - \sum\limits_{\substack{m\in I(i) \\ m\neq N}} f_{mi} = r_i & i = 1,\ldots,N-1 \\ f_{i\ell} \geq 0 \quad \forall (i,\ell)\in\mathcal{L} & \text{with } i\neq N. \end{cases}$$

If we define the total traffic at node i to be

$$t_i = r_i + \sum_{\substack{m\in I(i) \\ m\neq N}} f_{mi} \quad \text{for } i = 1,\ldots,N-1$$

we may then (for all i with $t_i \neq 0$) define the fraction of traffic at node i which will travel via link (i,ℓ) to be

$$\varphi_{i\ell} = \frac{f_{i\ell}}{t_i} \quad \text{when } (i,\ell)\in\mathcal{L}.$$

(SFP) may be reformulated in terms of these variables as in [2].

For each node $i\neq N$ fix an ordering of the links $(i,\ell)\in\mathcal{L}$ having $\ell\in\mathcal{O}(i)$ and let φ_i be the column vector with entries $\varphi_{i\ell}\ (i,\ell)\in\mathcal{L}$ so ordered. Identify with each set $\{\varphi_{i\ell} \mid (i,\ell)\in\mathcal{L} \quad i=1,\ldots,N-1\}$ the column vector $\varphi = (\varphi_1^T,\ldots,\varphi_{N-1}^T)$. Let

$$\bar{\Phi} = \{\varphi \mid \varphi_{i\ell} \geq 0 \text{ for all } (i,\ell) \in \mathcal{L}, \sum_{\ell \in \mathcal{O}(i)} \varphi_{i\ell} = 1 \text{ for } i=1,\ldots,N-1\}$$

and let

$$\Phi = \{\varphi \in \bar{\Phi} \mid \exists\ (i,\ell),\ldots,(m,N) \in \mathcal{L} \text{ forming a connected directed path from } i \text{ to } N \text{ for all } i=1,\ldots,N-1 \text{ with } \varphi_{i\ell} > 0,\ldots,\varphi_{mN} > 0\}.$$

One may easily verify that Φ and $\bar{\Phi}$ are convex and that the closure of Φ is $\bar{\Phi}$.

Gallager has shown (see [2], Theorem 1) that for each $\varphi \in \Phi$ and $r = (r_1,\ldots,r_{N-1})^T$ with $r_i \geq 0$ for all $i=1,\ldots,N-1$, there exist unique vectors $t(\varphi,r) = (t_1(\varphi,r),\ldots,t_{N-1}(\varphi,r))^T$ and $f(\varphi,r)$ with coordinates $f_{i\ell}(\varphi,r)$ $(i,\ell) \in \mathcal{L}$ $i \neq N$, ordered as was φ, which satisfy

$$t(\varphi,r) \geq 0 \qquad f(\varphi,r) \geq 0$$

$$t_i(\varphi,r) = r_i + \sum_{\substack{m \in I(i) \\ m \neq N}} f_{mi}(\varphi,r) \qquad \text{for } i=1,\ldots,N-1$$

$$\sum_{\ell \in \mathcal{O}(i)} f_{i\ell}(\varphi,r) - \sum_{\substack{m \in I(i) \\ m \neq N}} f_{mi}(\varphi,r) = r_i \quad \text{for } i=1,\ldots,N-1$$

$$f_{i\ell}(\varphi,r) = t_i(\varphi,r)\varphi_{i,\ell} \qquad i=1,\ldots,N,\ (i,\ell) \in \mathcal{L}.$$

Furthermore the functions $t(\varphi,r)$ and $f(\varphi,r)$ are twice continuously differentiable in the relative interior of $\Phi \times \{r \mid r \geq 0\}$, their common domain, and these derivatives can be extended to the relative boundary by taking the limit through the relative interior. Also for each $r \geq 0$ we have: given an f feasible for (SFP) there exists $\varphi \in \Phi$ with $f(\varphi,r) = f$.

It now follows that we may rewrite (SFP) in terms of the variables $\varphi_{i\ell}$ as

$$\begin{aligned}&\text{minimize } D(\varphi,r)\\ &\text{subject to } \varphi\in\Phi\end{aligned} \tag{4}$$

where $D(\varphi,r) = \sum_{\substack{(i,\ell)\in\mathcal{L}\\ i\neq N}} D_{i\ell}[f_{i\ell}(\varphi,r)]$.

If we ignore the difference between Φ and $\bar{\Phi}$ we see that (4) has the same form as the problem (1) considered earlier. Therefore we consider the iteration

$$\varphi_i^{k+1} = \bar{\varphi}_i^k \qquad i=1,\ldots,N-1 \tag{5}$$

where $\bar{\varphi}^k$ solves

$$\begin{aligned}&\text{minimize}\left(\frac{\partial D(\varphi^k,r)}{\partial\varphi_i}\right)^T(\varphi_i-\varphi_i^k) + \frac{1}{2}(\varphi_i-\varphi_i^k)^T M_i^k(\varphi_i-\varphi_i^k)\\ &\text{subject to } \varphi_i\geq 0,\quad \sum_{\ell\in\mathcal{O}(i)}\varphi_{i\ell}=1\end{aligned} \tag{6}$$

where $\frac{\partial D(\varphi^k,r)}{\partial\varphi_i}$ is the vector of the partial derivatives $\frac{\partial D(\varphi^k,r)}{\partial\varphi_{i\ell}}$, $\ell\in$ (i), evaluated at (φ^k,r). This is the gradient projection method (2) with M_k in (3) block diagonal form with $M_1^k,\ldots,M_{N-1}^k$ along the diagonal.

For a given $\varphi\in\Phi$ we define a node k to be <u>downstream</u> from a node i if there is a directed path from i to k such that for every link (ℓ,m) in that path we have $\varphi_{\ell m}>0$. In this case we also refer to i as being <u>upstream</u> from k. If there are no two distinct nodes i and k such that i is both upstream and downstream from k we say that φ is <u>loop-free</u>. Notice that φ is loop-free if and only if the downstream (or equivalently upstream) relation defines a partial ordering (see [5]) of $\mathcal{N}$ the set of nodes.

It is necessary to have $\varphi^{k+1}\in\Phi$ whenever $\varphi^k\in\Phi$, where φ^{k+1} is given by (5). We can ensure this by insisting that both φ^k and φ^{k+1} are loop-free. We will see later that loop-freedom will also facilitate efficient computation of the derivatives $\frac{\partial D}{\partial\varphi_{k\ell}}$ needed in (6).

For any $\varphi \in \Phi$ and $r \geq 0$ the partial derivatives $\frac{\partial D(\varphi,r)}{\partial \varphi_{i\ell}}$ may be computed from

$$\frac{\partial D}{\partial \varphi_{i\ell}} = t_i [D'_{i\ell}(f_{i\ell}) + \frac{\partial D}{\partial r_\ell}] \quad (i,\ell) \in \mathcal{L}, \quad i = 1,\ldots,N-1, \tag{7}$$

where $f_{i\ell}$ is the flow on (i,ℓ) which is uniquely defined for each $(i,\ell) \in \mathcal{L}$ by φ and r, $D'_{i\ell}(f_{i\ell})$ denotes the first derivative of $D_{i\ell}$ with respect to $f_{i\ell}$ and $\frac{\partial D}{\partial r_\ell}$ is given by

$$\frac{\partial D}{\partial r_\ell} = \sum_m \varphi_{\ell m} [D'_{\ell m}(f_{\ell m}) + \frac{\partial D}{\partial r_m}] \quad i = 1,\ldots,N-1 \tag{8}$$

$$\frac{\partial D}{\partial r_N} = 0. \tag{9}$$

These equations uniquely determine $\frac{\partial D}{\partial \varphi_{i\ell}}$ and $\frac{\partial D}{\partial r_i}$. For a more detailed discussion of the above ideas and derivations of equations (7), (8), and (9) see [2] and [1].

The algorithm (5),(6) must be modified so that loop-freedom is maintained. Certain of the variables $\varphi_{i\ell}$, which must be kept at zero in order to maintain loop-freedom, are specified in the following definition [1].

Definition: For a $\varphi \in \Phi$ and $i=1,\ldots,N-1$ the set of blocked nodes for φ at i, $B(i,\varphi)$, is the set of all $\ell \in \mathcal{J}(i)$ with $\varphi_{i\ell} = 0$ such that $\frac{\partial D(\varphi,r)}{\partial r_i} < \frac{\partial D(\varphi,r)}{\partial r_\ell}$ or there exists a link (m,n) with n downstream from ℓ and $\frac{\partial D(\varphi,r)}{\partial r_m} < \frac{\partial D(\varphi,r)}{\partial r_n}$ (a link such as (m,n) is referred to as improper).

We may now state a modified version of (5),(6):

$$\varphi_i^{k+1} = \bar{\varphi}_i^k \tag{10}$$

where $\bar{\varphi}_i^k$ is any solution of the problem

$$\text{minimize } \delta_i(\varphi^k,r)^T(\varphi_i - \varphi_i^k) + \frac{1}{2}(\varphi_i - \varphi_i^k)^T M_i^k (\varphi_i - \varphi_i^k)$$

$$\text{subject to } \varphi_i \geq 0, \ \sum_\ell \varphi_{i\ell} = 1, \ \varphi_{i\ell} = 0, \quad \forall \ell \in B(i,\varphi^k) \tag{11}$$

and where the vector $\delta_i(\varphi^k,r)$ has components

$$\delta_{i\ell}(\varphi^k,r) = D'_{i\ell}[f_{i\ell}(\varphi^k,r)] + \frac{\partial D(\varphi^k,r)}{\partial r_\ell}.$$

For each i with $t_i(\varphi^k,r) > 0$, the matrix M_i^k is some symmetric matrix which is positive definite on the subspace

$$\{v_i \mid \sum_{\ell \notin B(i,\varphi^k)} v_{i\ell} = 0\},$$

but $M_i^k = 0$ for those with $t_i(\varphi^k,r) > 0$.

Some important properties of the algorithm (10),(11) are given in the following proposition (this is Proposition 1 in [1] and its proof may be found there).

Proposition: a) If φ^k is loop-free then φ^{k+1} is loop-free.

b) If φ^k is loop-free and solves problem (11) then φ^k is optimal [for (4)].

c) If φ^k is optimal then φ^{k+1} is also optimal.

d) If $\bar{\varphi}_i^k \neq \varphi_i^k$ for some i for which $t_i(\varphi^k,r) > 0$ then there exists a positive scalar $\hat{\eta}_k$ such that

$$D[\varphi^k + \eta(\bar{\varphi}^k - \varphi^k),r] < D(\varphi^k,r) \qquad \forall \eta \in (0,\hat{\eta}_k].$$

It should be noted that (11) was derived from (6) by dividing the objective function in (6) by $t_i(\varphi^k,r)$ and assimilating the factor $[t_i(\varphi^k,r)]^{-1}$ into M_i^k.

2.1.3. Selection and Calculation of M_i^k

As mentioned earlier letting M_i^k be the Hessian of $D(\varphi^k,r)$ with respect to φ gives excellent convergence. We attempt to approximate this success by letting M_i^k be a diagonal matrix with $[t_i(\varphi^k,r)]^{-1}[\frac{\partial^2 D(\varphi^k,r)}{\partial \varphi_{i\ell}^2}]$ $(i,\ell) \in \mathcal{L}$ along the diagonal. These second derivatives are very hard to compute but reasonably accurate upper and lower bounds for them have been calculated [1].

For all ℓ

$$\underline{R}_\ell < \frac{\partial^2 D}{(\partial r_\ell)^2} \leq \overline{R}_\ell$$

where $\underline{R}_\ell$ and $\overline{R}_\ell$ are generated by

$$\underline{R}_\ell = \sum_m \varphi_\ell^2 (D''_{\ell m} + \underline{R}_m)$$
$$\overline{R}_\ell = \sum_m \varphi_{\ell m}^2 D''_{\ell m} + [\sum_m \varphi_{\ell m}\sqrt{\overline{R}_m}]^2$$
$$\underline{R}_N = \overline{R}_N = 0.$$

These are used to calculate corresponding bounds on $\frac{\partial^2 D}{\partial \varphi_{i\ell}^2}$ $\ell \notin B(i,\varphi)$ via

$$\underline{\Phi}_{i\ell} = t_i^2(D''_{i\ell} + \underline{R}_\ell)$$
$$\overline{\Phi}_{i\ell} = t_i^2(D''_{i\ell} + \overline{R}_\ell). \tag{12}$$

2.1.4. The Algorithm

We arrive at the following algorithm. If $t_i = 0$, we take M_i^k in (11) to be zero, and if $t_i \neq 0$ we take M_k^k to be the diagonal matrix with $\overline{\Phi}_{i\ell}$, $\ell \in O(i)$, along the diagonal. With this choice of M_i^k (for $t_i \neq 0$) the subproblem (11) may be rewritten as

$$\text{minimize } \sum_\ell \{\delta_{i\ell}(\varphi_{i\ell} - \varphi_{i\ell}^k) + \frac{\overline{\Phi}_{i\ell}}{2t_i}(\varphi_{i\ell} - \varphi_{i\ell}^k)^2\}$$
$$\text{subject to } \varphi_{i\ell} \geq 0, \sum_\ell \varphi_{i\ell} = 1, \varphi_{i\ell} = 0 \quad \forall \ell \in B(i,\varphi^k) \tag{13}$$

and solved using a Lagrange multiplier technique. By substituting for $\overline{\Phi}_{i\ell}$ the expression in (12) and turning the crank we obtain

$$\overline{\varphi}_{i\ell}^k = \max\{0, \varphi_{i\ell}^k - \frac{\delta_{i\ell} - \lambda}{t_i(D''_{i\ell} + \overline{R}_\ell)}\} \tag{14}$$

where λ is a Lagrange multiplier determined from

$$\sum_{\ell \notin B(i,\varphi^k)} \max\{0, \varphi_{i\ell}^k - \frac{\delta_{i\ell} - \lambda}{t_i(D''_{i\ell} + \overline{R}_\ell)}\} = 1 \tag{15}$$

which is piecewise linear in λ. We see from (14) that all $\varphi_{i\ell}$ with $\delta_{i\ell} < \lambda$ will be increased or remain one, while all those with $\delta_{i\ell} > \lambda$ will be decreased or remain zero.

2.2. Linesearches

An obvious candidate for a linesearch is to minimize along the direction of descent obtained by solving (11), i.e. let

$$\varphi^{k+1} = \varphi^k + \eta_k(\bar{\varphi}^k - \varphi^k) \tag{16}$$

where $\bar{\varphi}^k$ solves (11) and

$$\eta_k = \arg[\min_{\eta \in S_k} D[\varphi^k + \eta(\bar{\varphi}^k - \varphi^k), r]] \tag{17}$$

where $S_k = \{\eta \mid \varphi^k + \eta(\bar{\varphi}^k - \varphi^k) \geq 0\}$.

Actually an even simpler choice is to search in the space of flows f rather than the space of routing variables φ, i.e. if $\bar{\varphi}^k$ solves (11), and f^k and $\bar{f}^k$ are the flows corresponding to φ^k and $\bar{\varphi}^k$ respectively, then $\forall (i,\ell) \in \mathcal{L}$ let

$$\varphi_{i\ell}^{k+1} = \begin{cases} \dfrac{f_{i\ell}^k(\eta_k)}{\sum_{m \in \mathcal{O}(i)} f_{im}^k(\eta_k)} & \text{if } \sum_{m \in \mathcal{O}(i)} f_{im}^k(\eta_k) > 0 \\ \bar{\varphi}_{i\ell}^k & \text{if } \sum_{m \in \mathcal{O}(i)} f_{im}^k(\eta_k) = 0 \end{cases} \tag{18}$$

where

$$\eta_k = \arg[\min\{D_T[f^k(\eta)] \mid f^k(\eta) \geq 0\}] \tag{19}$$

where we have used the notation

$$D_T(f) = \sum_{(i,\ell) \in \mathcal{L}} D_{i\ell}(f_{i\ell})$$

and

$$f^k_{i\ell}(\eta) = f^k_{i\ell} + \eta(\bar{f}^k_{i\ell} - f^k_{i\ell}), \quad \forall \eta \geq 0 \quad (i,\ell) \in \mathcal{L}.$$

We have that

$$\left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=0} = \sum_{(i,\ell)\in\mathcal{L}} D'_{i\ell}(f^k_{i\ell})(\bar{f}^k_{i\ell} - f^k_{i\ell})$$

$$= \sum_{(i,\ell)\in\mathcal{L}} t_i(\bar{\varphi}^k, r)\,\delta_{i\ell}(\varphi^k, r)(\bar{\varphi}^k_{i\ell} - \varphi^k_{i\ell}).$$

Therefore if there exists an i with $\bar{\varphi}^k_i \neq \varphi^k_i$ and $t_i(\bar{\varphi}^k, r) \neq 0$ we have from the necessary condition for optimality in (11) that

$$\left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=0} < 0.$$

Thus $(\bar{f}^k - f^k)$ is a direction of descent for $D_T(f)$ at f^k and (18)-(19) is a reasonable choice. Also $D_T[f^k(\eta)]$ is a convex function in η, this makes the minimization in (19) relatively easy to perform, and evaluating $D_T[f^k(\eta)]$ at specific points is much simpler than evaluating $D[\varphi^k + \eta(\bar{\varphi}^k - \varphi^k), r]$.

A problem that can arise with the algorithm (18)-(19) is that if for some $(i,\ell)\in\mathcal{L}$ (i,ℓ) is improper at the kth iteration (note that this implies $f^k_{i\ell} > 0$) and $\eta_k < 1$ then even if $\bar{f}^k_{i\ell} = 0$ we will still have $\varphi^{k+1}_{i\ell} > 0$. Thus it is likely that (i,ℓ) will remain improper. Of course, this will only be a significant problem if $\eta_k < 1$ for several consecutive iterations thereby impeding convergence.

The following algorithm has been proposed [1] to resolve this difficulty. Given a loopfree φ^k and $\eta \in (0,1]$ we denote by $\bar{\varphi}^k_i(\eta)$ a solution of the following altered version of (11)

$$\text{minimize } \delta_i(\varphi^k, r)^T(\varphi_i - \varphi^k_i) + \frac{1}{2\eta}(\varphi_i - \varphi^k_i)^T M^k_i(\varphi_i - \varphi^k_i)$$
$$\text{subject to } \varphi_i \geq 0,\ \sum_\ell \varphi_{i\ell} = 1,\ \varphi_{i\ell} = 0 \quad \forall \ell \in B(i;\varphi^k) \qquad (20)$$

with M_i^k positive definite on the subspace $\{v_i \mid \sum_{\ell \notin B(i,\varphi^k)} v_{i\ell} = 0\}$ if $t_i(\varphi^k,r) > 0$ and $M_i^k = 0$ otherwise. Fix $\beta \in (0,1)$ and $\sigma \in (0,0.5)$. The algorithm is

$$\varphi^{k+1} = \bar{\varphi}^k(\beta^{m_k}) \tag{21}$$

where m_k is the first nonnegative integer m satisfying

$$D(\varphi^k,r) - D[\bar{\varphi}^k(\beta^m),r] \geqslant \sigma \frac{\partial D(\varphi^k,r)^T}{\partial \varphi} [\varphi^k - \bar{\varphi}^k(\beta^m)] \tag{22}$$

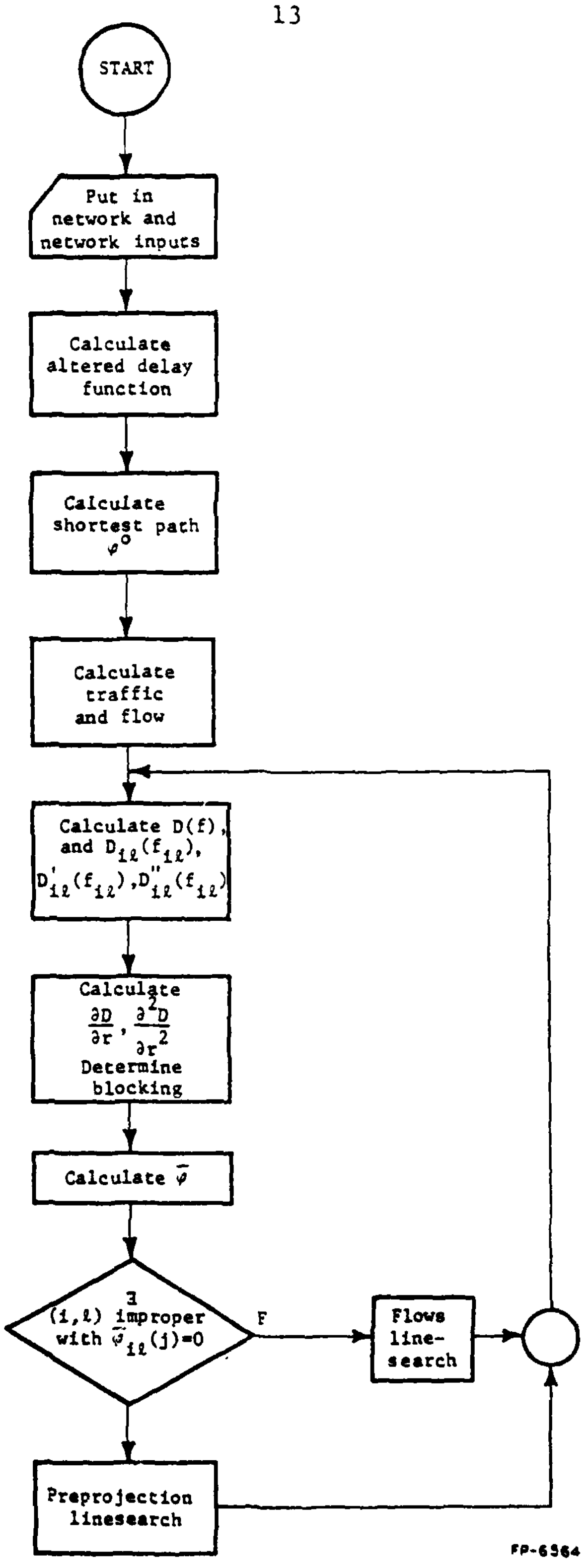
unless it is found that $\bar{\varphi}^k_{i\ell}(\beta^m) > 0$ for all (i,ℓ) which are improper for φ^k, then we switch to a search along the ray from $f(\varphi^k,r)$ through $f(\bar{\varphi}^k(\beta^m),r)$.

Although we do not prove convergence here, algorithms similar to this one have been shown to converge under mild assumptions ([1],[14]) and those proofs can be modified to work here.

The algorithm (20)-(22) is far more burdensome computationally than the algorithm (18)-(19). Therefore we consider this combined algorithm: if φ^k has an improper link (i,ℓ) with $\bar{\varphi}_{i\ell} = 0$ then we use (20)-(22), if not then we use (18)-(19). Also (20)-(22) is modified so that for each $m < m_k$ [i.e. m with $\bar{\varphi}^k(\beta^m)$ failing the test (22)] $\bar{\varphi}^k(\beta^m)$ is examined.

2.3. Description of the Program

The basic algorithm described in the preceding parts of this chapter and outlined in the flow chart in Figure 1 was programmed in SAIL (see [9],[10]), an ALGOL-like language for the PDP10. Actually the program was developed in several stages. First the algorithm of Section 2.1 was programmed, then just the flows linesearch was incorporated, and finally the combined linesearch algorithm of the preceding section was added.



START
Put in network and network inputs
Calculate altered delay function
Calculate shortest path φ^0
Calculate traffic and flow
Calculate $D(f)$, and $D_{i\ell}(f_{i\ell})$, $D'_{i\ell}(f_{i\ell})$, $D''_{i\ell}(f_{i\ell})$
Calculate $\frac{\partial D}{\partial r}$, $\frac{\partial^2 D}{\partial r^2}$ Determine blocking
Calculate $\bar{\varphi}$
$\exists$ (i,ℓ) improper with $\bar{\varphi}_{i\ell}(j)=0$
F
Flows line-search
Preprojection linesearch
FP-6564

Figure 1. The basic algorithm.

All three were programmed in two different versions. One, called the all-at-once version, consists of one iteration in which φ^{k+1} is calculated for all commodities using just the information from φ^{k}. The other, called the one-at-a-time version, consists of one iteration for each commodity receiving flow with the derivatives used to calculate φ^{k+1} for a particular commodity being recomputed immediately before the iteration for that commodity. For a comparison see Chapter 3, particularly the eleven node network example.

The more algorithmically complex aspects of the basic program are detailed in the subsequent sections of this chapter. Those portions not covered were programmed in a straightforward manner.

2.3.1. Initial Routing

For the algorithm described above we must have an initial feasible set of routing variables φ^{o}. We used the "shortest-path" routing i.e. for each node i the flow destined for node j is routed along the path from node i to node j that contains the fewest links. If multiple shortest paths exist then the flow is divided among them equally. We note that if node k lies on a shortest path from node i to node j then the portion of this path which runs from k to j is a shortest path from k to j, therefore this determines a set of routing variables $\varphi^{o}\in\overline{\Phi}$. In fact, $\varphi^{o}\in\Phi$ (i.e., φ^{o} is feasible) since by its definition we have from each node to the destination a path with φ^{o} nonzero along that path.

There are many available methods for finding the shortest paths between all pairs of nodes in a network (see Dreyfus [6], section 2).

None of these is more computationally efficient than one by Floyd [7] which was based on a procedure by Warshall [8] and is the one we have used here.

2.3.2. Calculating the Traffic and Flow

Given a set of inputs r and a loop-free set of routing variables $\varphi\in\Phi$, we wish to determine the corresponding traffic $t(\varphi,r)$ and flow $f(\varphi,r)$. From the definitions of Section 2.1 we have the following formuli

$$t_i(j) = r_i(j) + \sum_{\substack{m\in I(i)\\ m\neq j}} f_{mi}(j) \qquad \forall i\in\mathcal{N},\ j\in\mathcal{N} \tag{23}$$

$$f_{mi}(j) = t_m(j)\varphi_{mi}(j) \qquad \forall(m,i)\in\mathcal{L},\quad \forall j\in\mathcal{N}. \tag{24}$$

In Section 2.1 we saw that for a fixed commodity j the downstream relation due to a loop-free φ determines a partial ordering on $\mathcal{N}$. For any maximal node i (i.e. no other node is upstream from i), (23) and (24) become

$$t_i(j) = r_i(j) \tag{25}$$

$$f_{mi}(j) = 0 \qquad \forall m\in I(i). \tag{26}$$

Having calculated the traffic and flow at all the maximal nodes using (25) and (26), we would like to proceed to nodes which are immediately downstream from some maximal node and calculate their traffic and flow using (23) and (24). These nodes are easily found since $\mathcal{O}(i)$ is compiled and stored for each $i\in\mathcal{N}$ in the initial portion of the program, and we have that k is such a node if and only if $k\in\mathcal{O}(i)$ and $\varphi_{ik}(j) > 0$. The one problem with this procedure is that a node which is immediately downstream from one maximal node may be two or more nodes downstream from another maximal node. Therefore to calculate the traffic and flow at a non-maximal node using (23) and (24) we must wait until the traffic and flow have been calculated at all upstream neighbors. This we accomplish by use of two n-dimensional logical vectors, i.e. vectors with entries true or false. One vector indicates those nodes for which the traffic and flow have already been calculated. The other indicates those nodes not indicated in the first vector which are immediately downstream from at least one node which is indicated in the first vector. When a node in the second vector has all its upstream neighbors in the first vector, its traffic and flow are calculated using (23) and (24) and it is removed from the second vector and indicated in the first. Finally, all its downstream neighbors are indicated in the second. This procedure is iterated and we terminate when the destination node j is indicated in the first vector since it is the unique minimal node, i.e. it is downstream from all others.

2.3.3. Calculating $\frac{\partial D(\varphi,r)}{\partial r}$, $\overline{\frac{\partial^2 D(\varphi,r)}{\partial r^2}}$, and $B(i,\varphi)$, and Determining Improper Links

We assume now that r and φ are given and $t(\varphi,r)$ and $f(\varphi,r)$ have been calculated as in the preceding section. We recall the following formuli from Section 2.1

$$\frac{\partial D(\varphi,r)}{\partial r_i(j)} = \sum_{k\in\mathcal{O}(i)} \varphi_{ik}(j)[D'_{ik}(f_{ik}) + \frac{\partial D(\varphi,r)}{\partial r_k(j)}] \quad i\neq j$$

$$\frac{\partial D(\varphi,r)}{\partial r_j(j)} = 0 \qquad (27)$$

$$\left[\overline{\frac{\partial^2 D(\varphi,r)}{\partial r_i^2(j)}}\right] = \sum_{k\in O(i)} \varphi_{ik}^2(j)D''_{ik}(f_{ik}) + \left[\sum_{k\in O(i)} \varphi_{ik}(j)\sqrt{\overline{\frac{\partial^2 D(\varphi,r)}{\partial r_k^2(j)}}}\right]^2, \quad i\neq j$$

$$\left[\overline{\frac{\partial^2 D(\varphi,r)}{\partial r_j^2(j)}}\right] = 0. \tag{28}$$

We note for fixed i and j that,just as (23) and (24) only use information from upstream neighbors of i, the formuli (27) and (28) only use information from downstream neighbors of i. Therefore we utilize the same two-vector passing scheme except we pass upstream not downstream. Here the stopping condition is more complicated, in that we must check all maximal nodes rather than just the one minimal node as in the previous section.

In order to determine for a given destination j and node $i\neq j$ the set $B(i;\varphi)$ we must consider all nodes $k\in\mathcal{O}(i)$ with $\varphi_{ik}(j) = 0$ and determine which of these satisfy

$$\frac{\partial D(\varphi,r)}{\partial r_i(j)} \leq \frac{\partial D(\varphi,r)}{\partial r_k(j)}$$

or are upstream from an improper link. Thus along with each pass upstream of information we pass knowledge of an improper link existing downstream. After this pass the newly computed $\frac{\partial D(\varphi,r)}{\partial r_i(j)}$ is compared with $\frac{\partial D(\varphi,r)}{\partial r_k(j)}$ for each k with $\varphi_{ik}(j) > 0$ to determine if (i,k) is improper. If this is the case for any such k then i is noted as a node with an improper link downstream (also (i,k) is noted for later as an improper link, see the section on linesearches). For those k with $\varphi_{ik}(j) > 0$ we have that $i\in B(k;\varphi)$. This is intuitively clear since otherwise we might get a loop from i to k to i. More rigorously, $\varphi_{ik}(j) > 0$ and φ loop-free imply $\varphi_{ki}(j) = 0$ and we see that if (i,k) is not improper then $\frac{\partial D(\varphi,r)}{\partial r_i(j)} > \frac{\partial D(\varphi,r)}{\partial r_k(j)}$ and the first part of the definition of $i \in B(k;\varphi)$ is satisfied, otherwise (i,k) is the downstream improper link for node i necessary to satisfy the second part of this definition.

We would like to determine $B(i;\varphi)$ at the time that we pass information to node i. However this is not possible since $\frac{\partial D(\varphi,r)}{\partial r_k(j)}$ may not have been calculated for some $k \in \mathcal{O}(i)$ with $\varphi_{ik}(j) = 0$. Therefore once $\frac{\partial D(\varphi,r)}{\partial r_i(j)}$ has been calculated for all i, we go back over $\mathcal{O}(i)$ for each i checking for additional $k \in \mathcal{O}(i)$ which belong in $B(i;\varphi)$.

2.3.4. Solution of (13) Via (14) and (15)

Again assuming the destination j is fixed we may rewrite (15) as

$$\sum_{\ell \notin B(i;\varphi)} \max\{0, \frac{1}{d_{i\ell}(j)}(\lambda - u_{i\ell}(j))\} = 1 \tag{29}$$

where we have used the following notation

$$d_{i\ell}(j) = t_i(j)[D''_{i\ell}(f_{i\ell}) + \bar{R}_\ell(j)]$$

$$u_{i\ell}(j) = d_{i\ell}(j) - \varphi_{i\ell}(j) d_{i\ell}(j).$$

Since $d_{i\ell}(j) \geq 0$, (29) is actually

$$\sum_{\substack{\ell \notin B(i;\varphi) \\ \ell \text{ such that} \\ u_{i\ell}(j) < \lambda}} \frac{1}{d_{i\ell}(j)} (\lambda - u_{i\ell}(j)) = 1. \tag{30}$$

Let M be the number of nodes in the set $\mathcal{O}(i) \setminus B(i;\varphi)$ (the set of nodes ℓ with $\ell \in \mathcal{O}(i)$ and $\ell \notin B(i;\varphi)$). We rearrange this set so that $u_{i\ell_1}(j) \leq u_{i\ell_2}(j) \leq \cdots \leq u_{i\ell_M}(j)$. If we let $\bar{m}$ be the smallest positive integer such that

$$\sum_{m=1}^{\bar{m}} \frac{1}{d_{i\ell_m}(j)} (u_{i\ell_{\bar{m}}}(j) - u_{i\ell_m}(j)) > 1$$

then we see that if λ^* solves (30) then $(\lambda^*,1)$ lies on the line segment between $(u_{i\ell_{\bar{m}-1}}(j), W[u_{i\ell_{\bar{m}-1}}(j)])$ and $(u_{i\ell_{\bar{m}}}(j), W[u_{i\ell_{\bar{m}}}(j)]$ where

$$W[\lambda] = \sum_{m=1}^{M} \max\{0, \frac{1}{d_{i\ell_m}(j)} (\lambda - u_{i\ell_m}(j))\}.$$

Therefore

$$\lambda^* = \frac{u_{i\ell_{\bar{m}}}(j)[1-W[u_{i\ell_{\bar{m}-1}}(j)]] + u_{i\ell_{\bar{m}-1}}[W[u_{i\ell_{\bar{m}}}(j)] - 1]}{(W[u_{i\ell_{\bar{m}}}(j)] - W[u_{i\ell_{\bar{m}-1}}(j)])}.$$

Of course, it is possible that we might have $W[u_{i\ell_M}(j)] \leq 1$. In this case

$$\lambda^* = \frac{1 + \sum_{i=1}^{M} \frac{u_{i\ell_m}(j)}{d_{i\ell_m}(j)}}{\sum_{i=1}^{M} \frac{1}{d_{i\ell_m}(j)}}.$$

2.3.5. Line Searches

We wish to solve the problem (19) of Section 2.2. We reiterate the important fact that the objective function,

$$D_T[f^k(\eta)] = \sum_{(i,\ell)\in\mathcal{L}} D_{i\ell}[f_{i\ell}^k + \eta(\bar{f}_{i\ell}^k - f_{i\ell}^k)],$$

is convex in the variable η (note that for a given f we use the notation $f_{i\ell} = \sum_{j=1}^{n} f_{i\ell}(j)$). This simplifies and increases the effectiveness of the following minimization procedure which entails fitting a cubic polynomial to the objective function [11]. Also it guarantees that the local minimum that this procedure reaches is a global one (along the line of search).

Since $(\bar{f}^k - f^k)$ is a descent direction we can be sure that the minimizing η^* will be positive. Thus we first check whether

$\left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=1} = 0$. If it does then $\eta^* = 1$. If not then we see if

$\left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=1} > 0$. If this is the case then we know $\eta^* \in (0,1)$.

The third possibility is that $\left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=1} < 0$. This implies $\eta^* > 1$. Of course since the constraint set to the original problem is compact and convex there exists a unique $\hat{\eta} \geqslant 1$ such that $\eta > \hat{\eta}$ if and only if $f^k_{i\ell}(j) + \eta(\bar{f}_{i\ell}(j) - f_{i\ell}(j)) < 0$ for some $(i,\ell) \in \mathcal{L}$, $j \in \mathcal{N}$. If, for some $(i,\ell) \in \mathcal{L}$, $j \in \mathcal{N}$, we have $f^k_{i\ell}(j) > 0$ and $\bar{f}^k_{i\ell}(j) = 0$ then $\hat{\eta} = 1$ otherwise $\hat{\eta}$ is the solution of

$$\underset{\substack{\text{over all}\\ (i,\ell)\in\mathcal{L},\ j\in\mathcal{N}\ \text{with}\\ f_{i\ell}(j) > \bar{f}_{i\ell}(j)}}{\text{minimize}} \left[\frac{f^k_{i\ell}(j)}{f^k_{i\ell}(j) - \bar{f}^k_{i\ell}(j)}\right].$$

Thus, $\left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=1} < 0$ implies $\eta^* \in (1,\hat{\eta}]$. If $\left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=\hat{\eta}} < 0$ then $\eta^* = \hat{\eta}$ otherwise $\eta^* \in (0,\hat{\eta})$.

If we have not found η^* at this point (i.e. if $\eta^* \neq 1$ and $\eta^* \neq \hat{\eta}$) then we have determined an interval (A,B) (either (0,1) or $(1,\hat{\eta})$) such that $\eta^* \in (A,B)$. We fit a cubic polynomial to $D_T[f^k(A)]$, $D_T[f^k(B)]$, $\left.\frac{dD_T[f(\eta)]}{d\eta}\right|_{\eta=A}$, and $\left.\frac{dD_T[f(\eta)]}{d\eta}\right|_{\eta=B}$ and find the minimum of this polynomial which occurs at

$$C = B - \left[\frac{D'_T[f^k(B)] + W - Z}{D'_T[f(B)] - D'_T[f(A)] + 2W}\right](B-A)$$

where

$$D'_T[f^k(\cdot)] = \left.\frac{dD_T[f^k(\eta)]}{d\eta}\right|_{\eta=(\cdot)}$$

$$Z = 3\left[\frac{D_T[f^k(A)] - D_T[f^k(B)]}{B-A}\right] + D'_T[f^k(A)] + D'_T[f^k(B)]$$

and

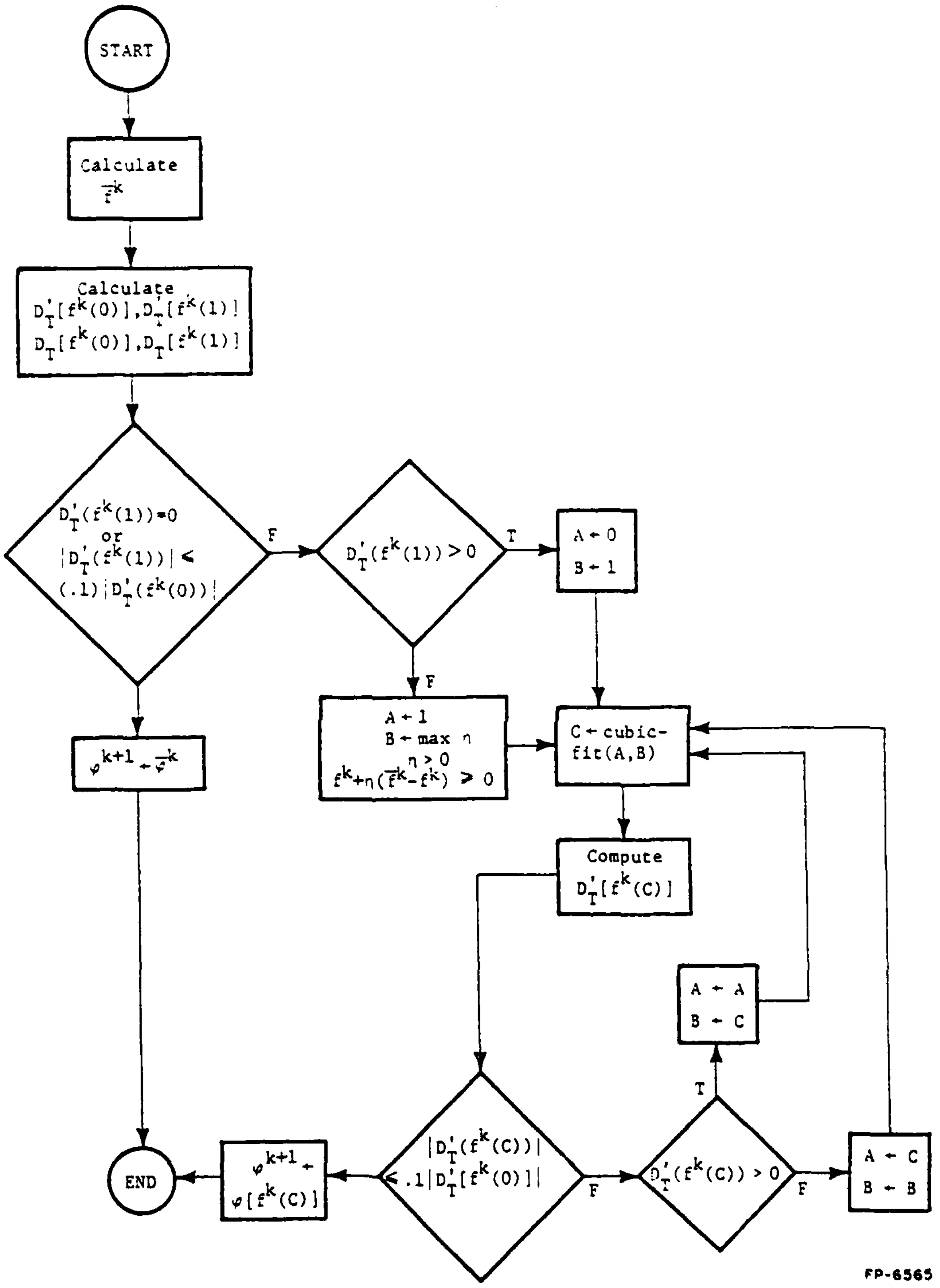
$$W = [Z^2 - D'_T[f^k(A)]D'_T[f^k(B)]]^{1/2}.$$

Next we evaluate $D'_T[f(C)]$. If $D'_T[f(C)] = 0$ then $\eta^* = C$. If $D'_T[f(C)] > 0$ then $\eta^* \in (A,C)$ so we let $B = C$. If $D'_T[f(C)] < 0$ then $\eta^* \in (C,B)$ so we let $A = C$.

In either of the last two cases we then repeat this procedure for the new (A,B).

Since the above algorithm may never terminate, we replaced the condition $D_T'[f(C)] = 0$ by $|D_T'[f(C)]| < \varepsilon|D_T'[f(0)]|$ where ε is some small positive number (we found $\varepsilon=.1$ appropriate). A flow chart for this algorithm may be found in Figure 2.

The other parts of the line search algorithm were programmed as presented in Section 2.2. We used $\beta = .5$ and $\sigma = .1$ in the Armijo rule (21)-(22).



START
Calculate $\bar{f}^k$
Calculate $D_T'[f^k(0)], D_T'[f^k(1)]$ $D_T[f^k(0)], D_T[f^k(1)]$
$D_T'(f^k(1))=0$ or $|D_T'(f^k(1))| < (.1)|D_T'(f^k(0))|$
F
$D_T'(f^k(1)) > 0$
T
$A \leftarrow 0$ $B \leftarrow 1$
F
$A \leftarrow 1$ $B \leftarrow \max_{\eta > 0} \eta$ $f^k+\eta(\bar{f}^k-f^k) \geq 0$
$C \leftarrow$ cubic-fit(A,B)
$\varphi^{k+1} \leftarrow \bar{\varphi}^k$
Compute $D_T'[f^k(C)]$
$A \leftarrow A$ $B \leftarrow C$
T
$|D_T'(f^k(C))| < .1|D_T'[f^k(0)]|$
F
$D_T'(f^k(C)) > 0$
F
$A \leftarrow C$ $B \leftarrow B$
$\varphi^{k+1} \leftarrow \varphi[f^k(C)]$
END
FP-6565

Figure 2. Flows line search.

CHAPTER 3

COMPUTATIONAL RESULTS FOR THE KLEINROCK DELAY FUNCTION

The programs referred to in Section 2.3 were implemented using a delay function based on the formula for queuing delay in an M/M/1 queue (see Kleinrock [15])

$$Q_{i\ell}(f) = \frac{f}{C_{i\ell}-f} \quad , \; f \in [0, C_{i\ell})$$

where for each $(i,\ell) \in \mathcal{L}$, $C_{i\ell}$ is a positive number called the capacity of (i,ℓ). In order to satisfy the defining properties from Chapter 2 (in particular that $D_{i\ell}$ be twice continuously differentiable on $[0,\infty)$) we used the following extended version of $Q_{i\ell}(f)$:

$$D_{i\ell}(f) = \begin{cases} Q_{i\ell}(f) & \text{if} \quad f \leq .99C_{i\ell} \\ Q_{i\ell}(.99C_{i\ell}) + Q'_{i\ell}(.99C_{i\ell})[f-.99C_{i\ell}] & \\ \qquad \frac{1}{2} Q''_{i\ell}(.99C_{i\ell})[f-.99C_{i\ell}]^2 & \text{if} \quad f > .99C_{i\ell} \, . \end{cases}$$

We note that not only does this meet all the requirements of Chapter 2 but if $f_{i\ell} \leq .99C_{i\ell}$, for all $(i,\ell) \in \mathcal{L}$, at the optimum then minimizing $\sum_{(i,\ell)\in\mathcal{L}} D_{i\ell}$ is equivalent to minimizing $\sum_{(i,\ell)\in\mathcal{L}} Q_{i\ell}$.

The various versions of the program were tested on several network configurations and most of these for many different inputs. We now present a selection of these results.

<u>8-node network</u>: The programs were run for the network in Figure 3 with inputs $r_i(j) = 3$ for $i = 1,\ldots,8$, and $j = [(i+1)(\text{modulo } 8)]+1$ or $j = [(i+5)(\text{modulo } 8)]+1$, and $r_i(j) = 0$ for all other (i,j). The results are listed in Table 1. (Note the following abbreviations: l.s. for line search, a-a-o for all-at-once, and o-a-a-t for one-at-a-time.)

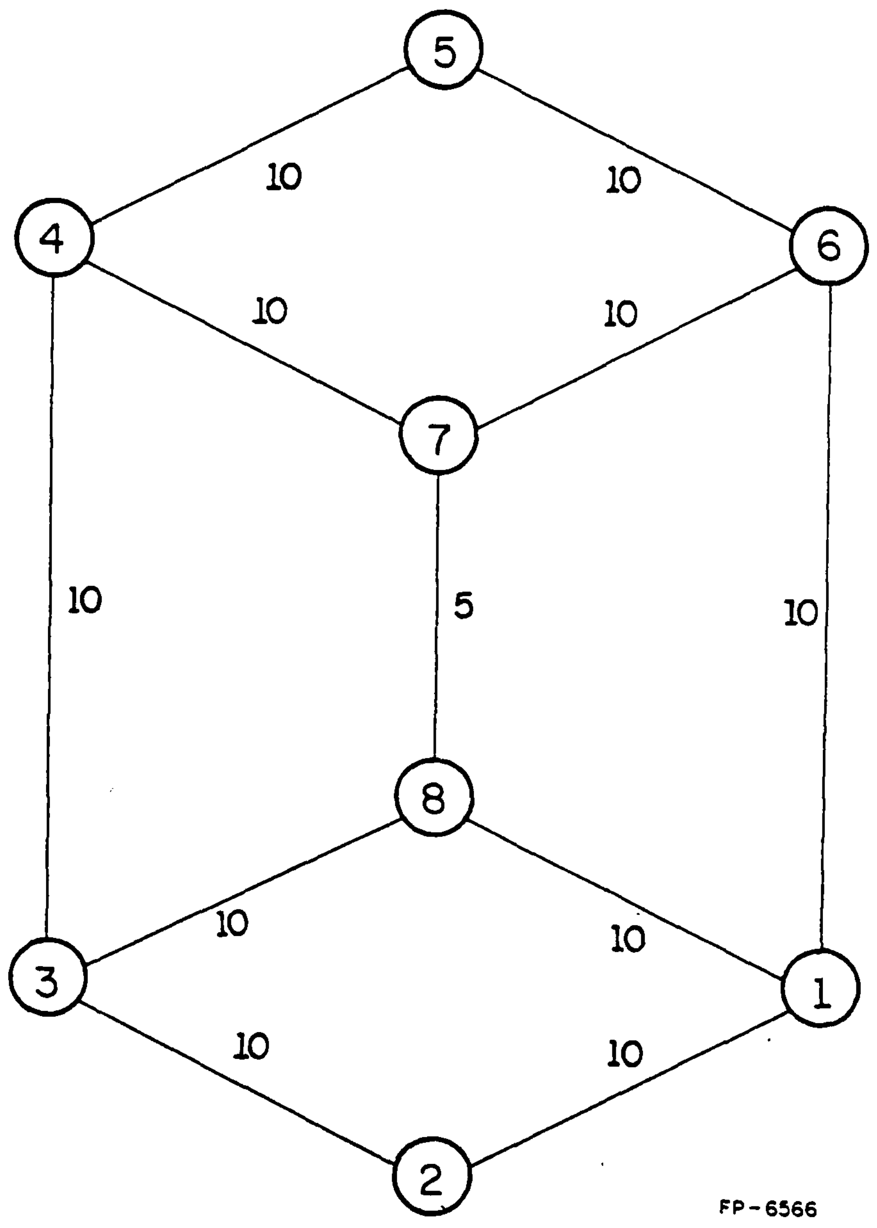


Figure 3. 8-node network.

Table 1. 8-node Network

Iteration	w/o l.s.	l.s. a-a-o	l.s. o-a-a-t
0	85,216.7	85,216.7	85,216.7
1	200.840	45.5844	200.840
2	63.2707	44.3713	48.9063
3	46.5705	43.6303	46.0621
4	44.8815	43.1652	45.0389
5	44.1566	42.8464	44.3377
6	43.7075	42.5588	43.7518
7	43.3439	42.3653	43.3220
8	43.0258	42.2802	43.0260
9	42.7485	42.1404	42.7940
10	42.5163	42.1059	42.6064
11	42.3418	42.0708	42.4301
12	42.2280	42.0462	42.2758
13	42.1878	42.0268	42.1784
14	42.1464	42.0122	42.1067
15	42.1094	42.0006	42.0487

The lowest value obtained for the objective function is 41.9593 which was achieved in the 104th iteration of the all-at-once algorithm with line search. After this iteration the maximum utilization, i.e. $\max\limits_{(i,\ell)\in\mathcal{L}} \frac{f_{i\ell}}{c_{i\ell}}$ (see Chapter 4), was approximately 0.82. The last improper link was eliminated in iteration 7 of the all-at-once with line search and iteration 15 of the one-at-a-time.

<u>11-node network</u>: The computational results for the network in Figure 4 are shown in Table 2. The inputs in this case are $r_6(8) = 5$, $r_1(8) = r_2(8) = r_4(8) = r_5(8) = 2$, $r_3(11) = 2$, and $r_i(j) = 0$ for all other (i,j). This somewhat contrived example was designed to show an essential difference between the all-at-once algorithms and the one-at-a-time with line search algorithm. The objective function value at the optimum was calculated to be 37.1542 with six significant digits. At this value the maximum utilization is approximately 0.75. From Table 2 we see that the one-at-a-time with line search algorithm converges in six iterations. The all-at-once with line search converges very slowly; in fact, it steadily decreased for one thousand iterations to 37.4961. And the all-at-once without line search actually oscillates and never converges. This unusual behavior is due to the fact that the two destinations (nodes 8 and 11) require different stepsizes for their respective iterations. The algorithm without line search consistently compromises, using a stepsize that is too large for the destination 8 iteration and too small for the destination 11 iteration. The all-at-once with line search merely chooses the best possible of the aforementioned compromises, and since the basic algorithm always provides a direction of descent (see Chapter 2), this version of of the algorithm will always lead to a decrease of the objective function,

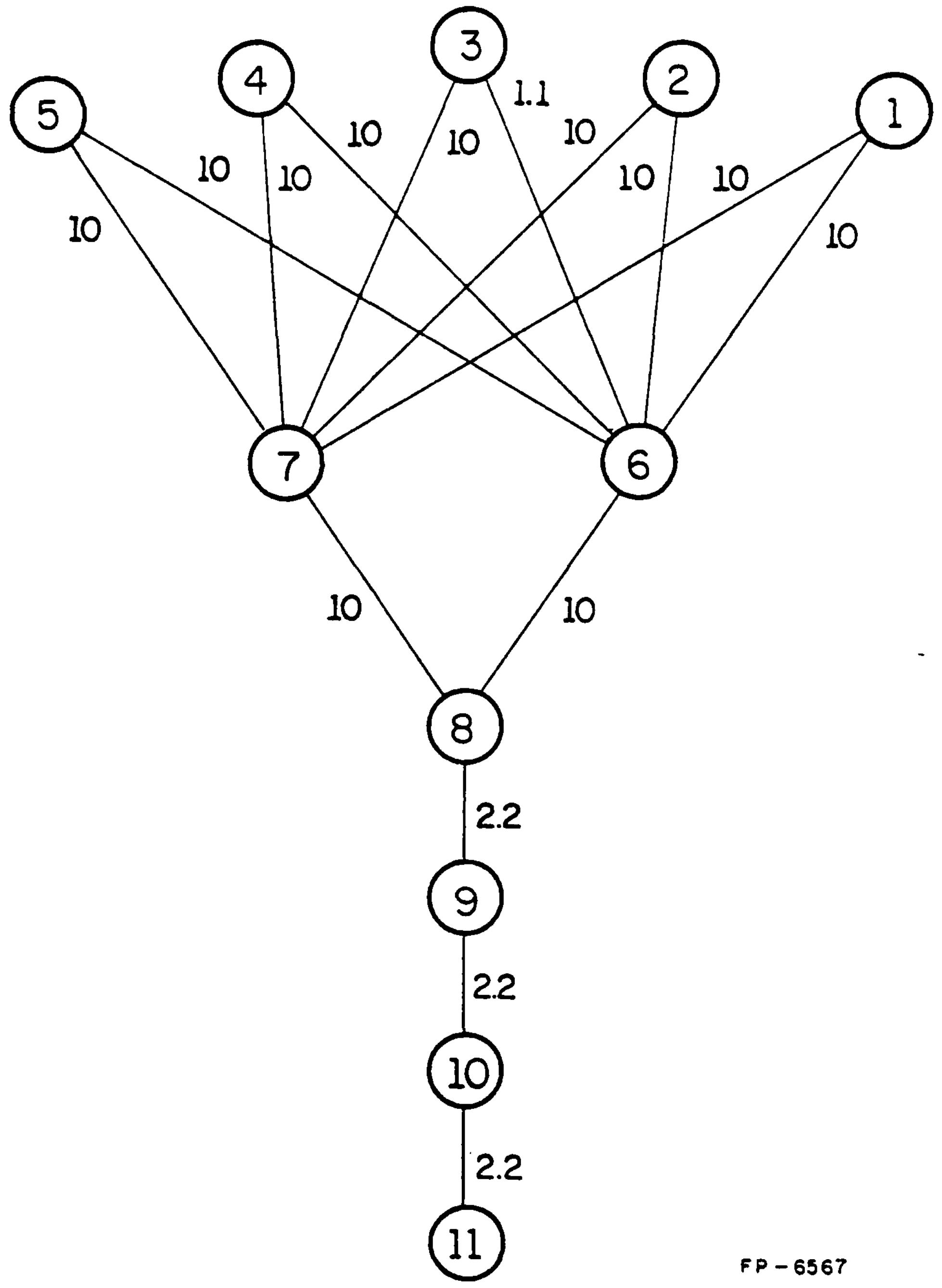


Figure 4. 11-node network.

Table 2. 11-node Network

Iteration	w/o l.s.	l.s. a-a-o	l.s. o-a-a-t
0	340.992	340.992	340.992
1	55.1392	49.1647	55.1392
2	44.3970	41.1710	37.9472
3	46.3674	40.6925	37.3075
4	43.4833	40.6341	37.2C43
5	45.7913	40.5629	37.1543
6	42.6035	40.5021	37.1542
7	44.9916	40.4442	37.1542
8		40.3921	
9		40.3421	
10		40.2960	
11		40.2517	
12		40.2102	
13		40.1702	
14		40.1325	
15		40.0960	

but, as we see, convergence may come very slowly. Of course, none of this causes any problem for the one-at-a-time with line search algorithm which simply chooses the appropriate stepsize for each destination and since there is little interaction between the flow destined for node 8 and that for node 11, the convergence is rapid.

16-node network: For the network in Figure 5 the computational results are given in Table 3 for the inputs $r_i(j) = 3$ for all $i = 1,\ldots,16$ and $j = [(i+3)(\text{mod } 16)]+1$ or $j = [(i+11)(\text{mod } 16)]+1$ and $r_i(j) = 0$ for all other (i,j). The smallest value obtained for the objective function is 69.1432. This was achieved after 55 iterations of the one-at-a-time line search algorithm by a routing with maximum utilization of about 0.76. The last improper links were removed in iteration 15 for the all-at-once with line search and in iteration 18 for the one-at-a-time.

φ^0, the initial point used here happened to be an inflection point, which accounts for the constancy of the value of the first iteration. This unusual situation did not come unpredicted, see [2], pp. 76-77.

26-node network: This network, shown in Figure 6 with results in Table 4 had inputs $r_i(j) = 5.5$ for $i = 1,\ldots,26$, $j = [(i+6)(\text{mod } 26)]+1$ or $j = [(i+19) \text{ mod } 26]+1$, and $r_i(j) = 0$ for all other (i,j). The lowest value achieved is 65.3304 after 40 iterations of the all-at-once line search algorithm where the maximum utilization was found to be about 0.75. The last improper link appeared in the 15th iteration of both algorithms.

32-node network: This network with results in Table 5 is shown in Figure 7. The inputs were $r_i(j) = 12$ for all $i = 1,\ldots,32$ and $j = [(i+7)(\text{mod } 32)]+1$ or $j = [(i+23)(\text{mod } 32)]+1$, and $r_i(j) = 0$ for all other (i,j). The smallest

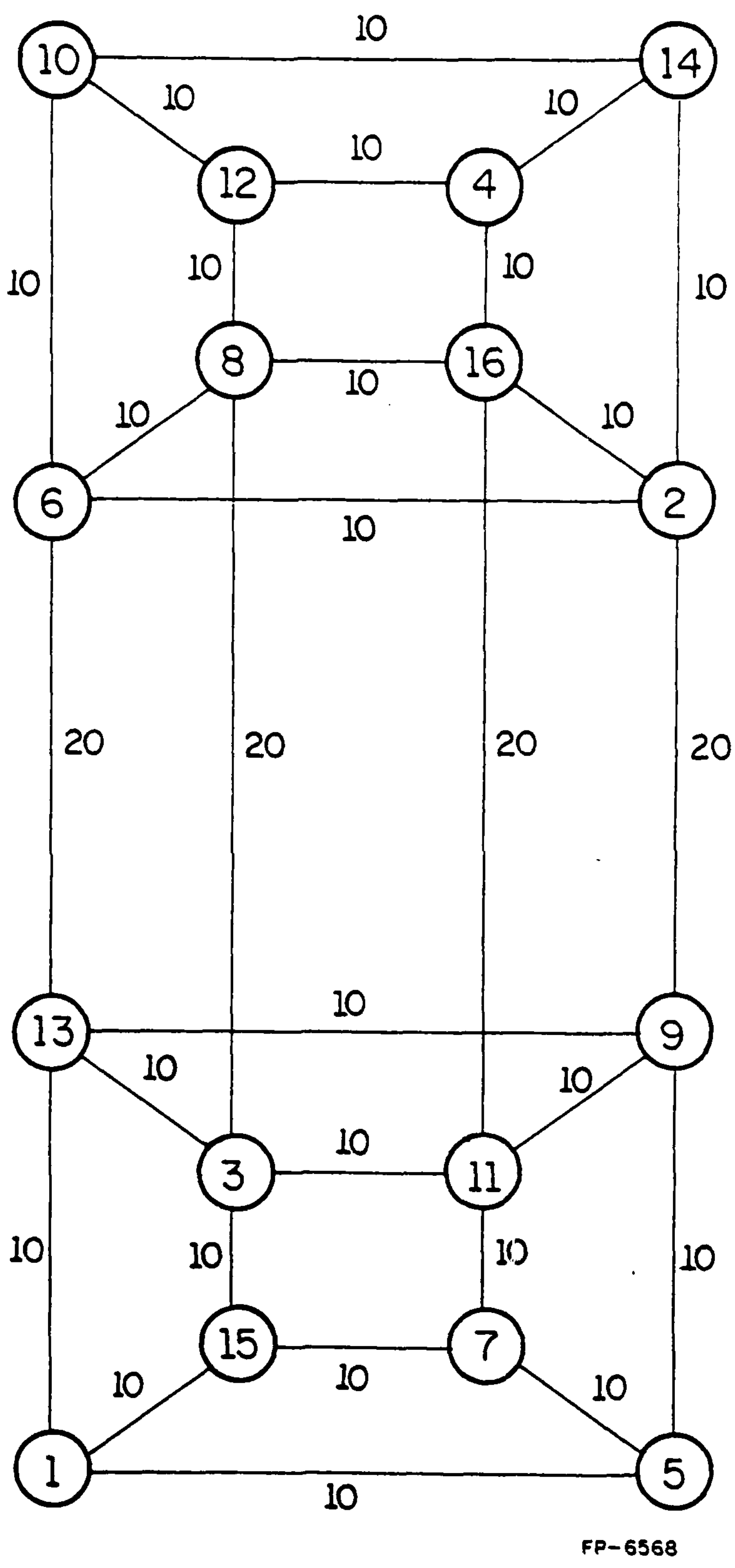
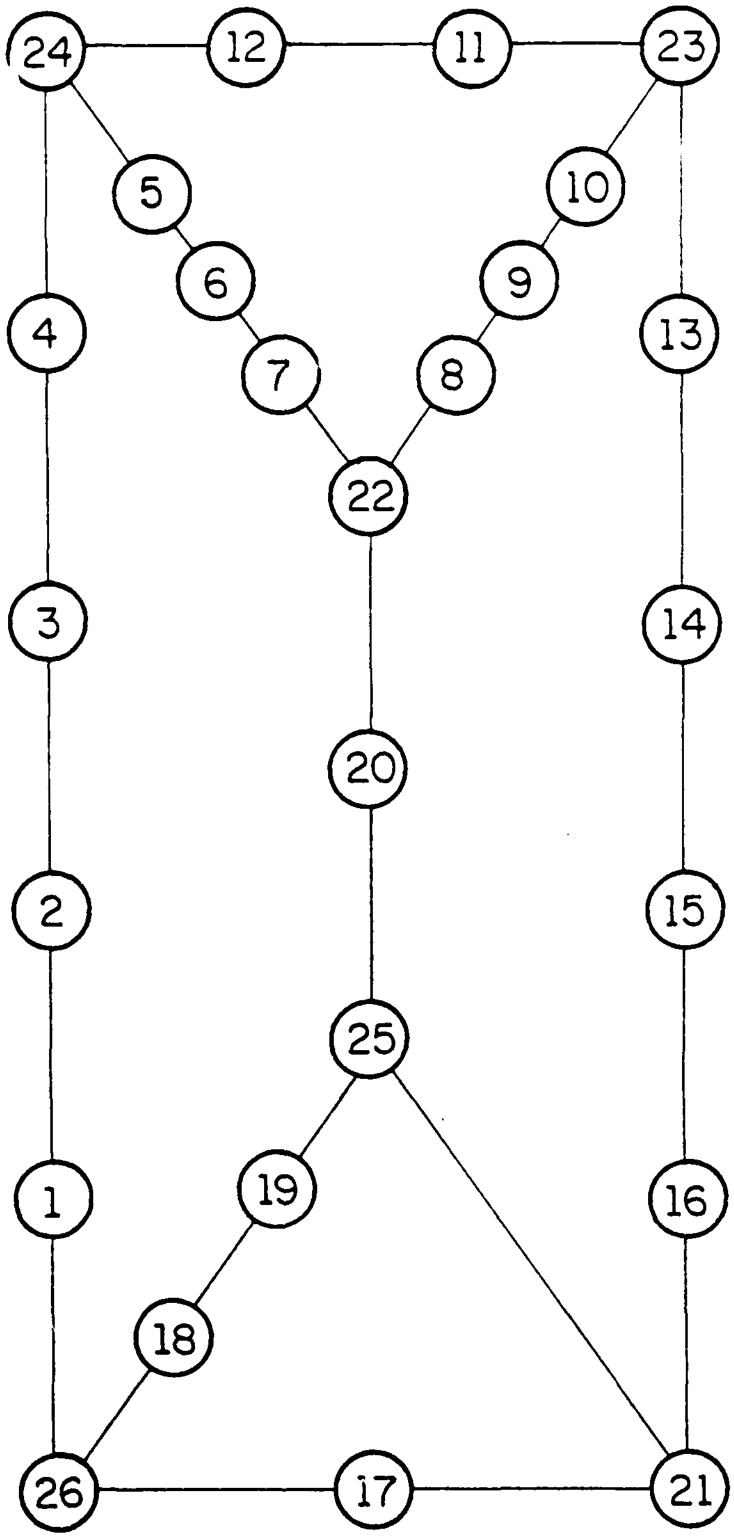


Figure 5. 16-node network.

Table 3. 16-node Network

Iteration	w/o l.s.	l.s. a-a-o	l.s. o-a-a-t
0	2415.93	2415.93	2415.93
1	2415.93	2415.93	2415.93
2	241.869	197.516	89.6904
3	99.5542	95.5172	73.1810
4	70.9437	69.7513	71.3961
5	70.5455	69.5667	70.6898
6	69.9786	69.4649	70.1559
7	69.7682	69.3996	69.7715
8	69.4698	69.3600	69.5746
9	69.3885	69.3332	69.4441
10	69.2789	69.3050	69.3668
11	69.2503	69.2745	69.3116
12	69.2256	69.2453	69.2631
13	69.1959	69.2276	69.2143
14	69.1620	69.2082	69.1922
15	69.1492	69.1890	69.1721



FP-6569

Figure 6. 26-node network (all capacities = 50).

Table 4. 26-node Network

Iteration	w/o l.s.	l.s. a-a-o	l.s. o-a-a-t
0	3840.44	3840.44	3840.44
1	92.1412	76.0794	74.3970
2	79.2213	70.0616	68.2911
3	73.4412	67.5766	67.0121
4	68.7712	66.9592	66.4396
5	67.9364	66.5555	66.2142
6	67.2223	66.2944	66.1112
7	66.7473	66.2457	66.0541
8	66.4142	66.0359	65.9321
9	66.0317	65.9549	65.8197
10	65.9131	65.8992	65.7648
11	65.8552	65.7897	65.6613
12	65.7893	65.6702	65.5172
13	65.6795	65.6300	65.5551
14	65.5545	65.5444	65.4544
15	65.4632	65.4743	65.3912

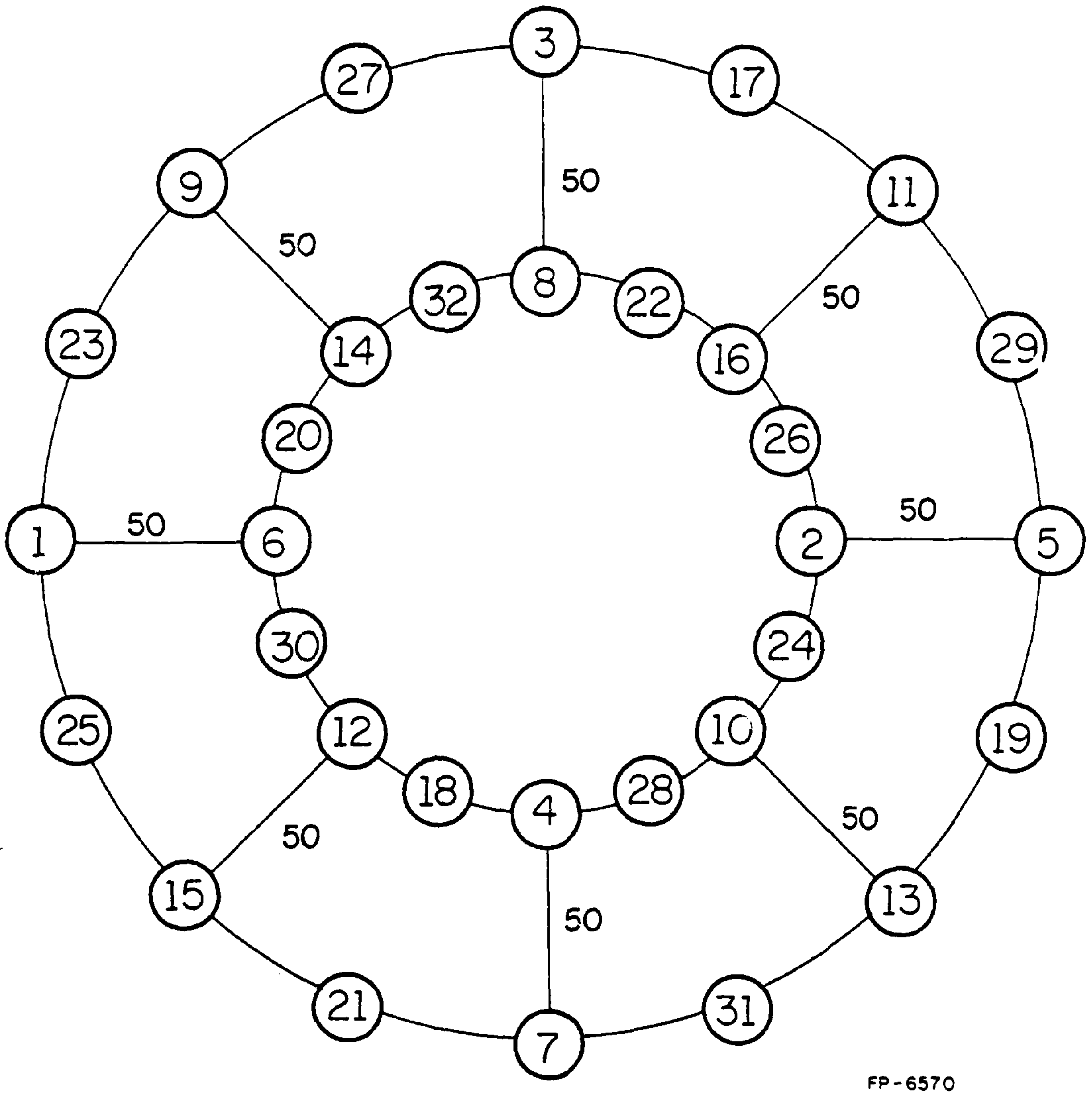


Figure 7. 32-node network (all unlabeled nodes have capacity 30).

Table 5. 32-node Network

Iteration	w/o l.s.	l.s. a-a-o	l.s. o-a-a-t
0	4827.75	4827.75	4827.75
1	654.959	566.323	654.959
2	219.781	183.398	118.723
3	107.855	105.179	102.377
4	104.116	100.514	98.4962
5	98.5500	98.9765	96.6925
6	97.2813	98.3628	95.7593
7	96.8288	97.1440	95.2549
8	96.2908	96.8543	94.9314
9	95.7999	96.3476	94.8542
10	95.4387	95.9965	94.8240
11	94.9667	95.5426	94.8008
12	94.8650	95.2509	94.7918
13	94.8212	95.0177	94.7843
14	94.8021	94.8764	94.7791
15	94.8168	94.8309	94.7762

value obtained was 94.7658, which was achieved after 39 iterations of the all-at-once line search algorithm. At this point the maximum utilization was 0.61. The last improper links were found in iteration 7 of the one-at-a-time line search algorithm and iteration 15 of the all-at-once line search algorithm.

61-node network: This network, shown in Figure 8, has the ARPANET topology as of August 1978. The results are given in Table 6 and the inputs are $r_i(j) = 3.75$ for $i = 1,\ldots,61$ and $j = [(i+14)(\mathrm{mod}\ 61)]+1$ or $j = [(i+45)(\mathrm{mod}\ 61)]+1$, and $r_i(j) = 0$ otherwise. 15 iterations of the one-at-a-time algorithm and 73 iterations of the all-at-once line search algorithm were performed, in neither case were all improper links eliminated. The smallest objective function value was obtained in the all-at-once line search algorithm at 149.782 and the corresponding maximum utilization was about 0.87.

10- and 50-node networks: These networks, shown in Figures 9 and 10, respectively, are included to illustrate a point discussed in Chapter 4. The computational results are contained in Table 7 for both networks. The 10-node network has inputs $r_{10}(1) = r_3(1) = 20$ and $r_i(j) = 0$ for all other (i,j). The 50-node network has inputs $r_{50}(1) = r_3(1) = 20$ and $r_i(j) = 0$ for all other (i,j).

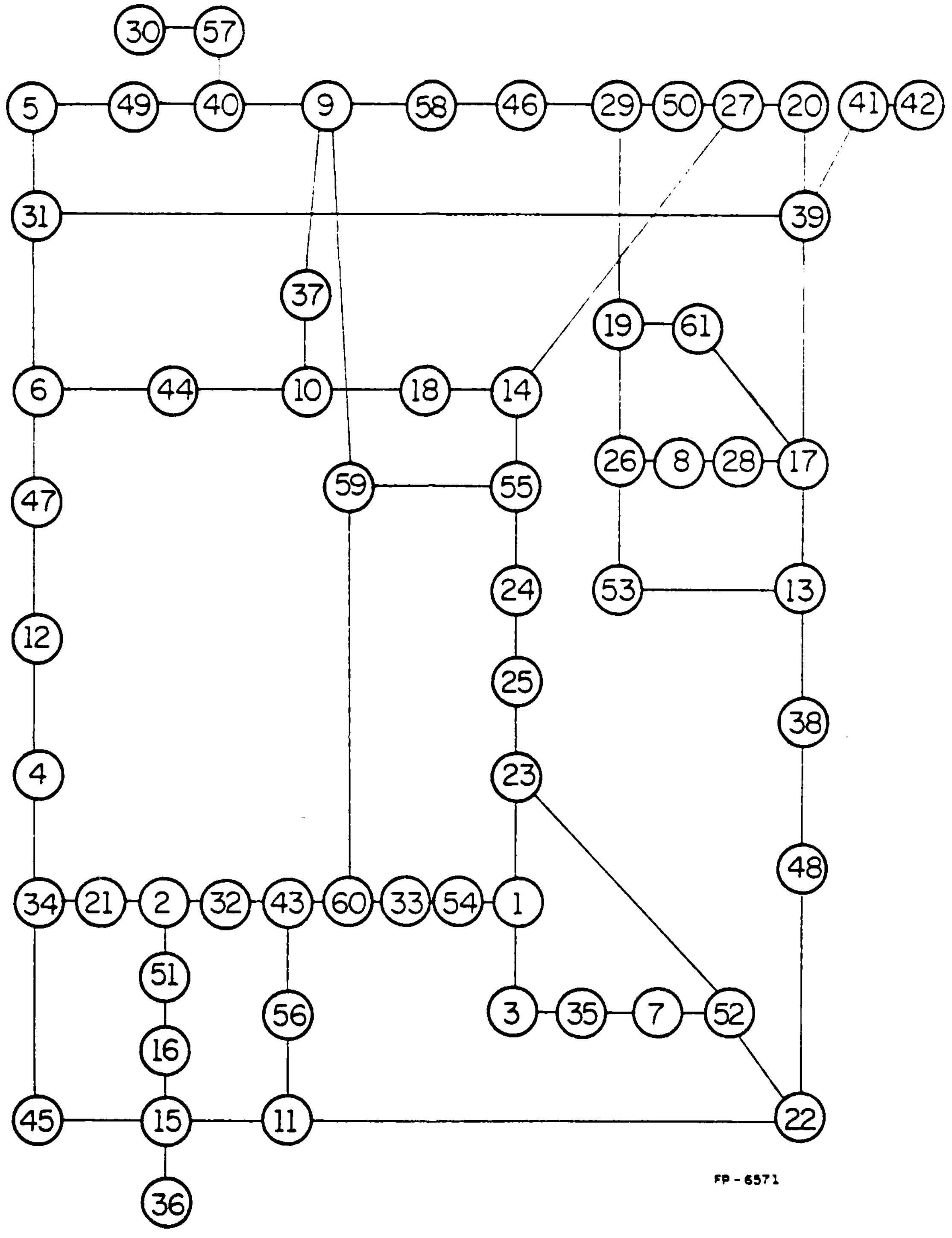


Figure 8. 61-node network (ARPANET topology as of 9/73)(all capacities = 50).

Table 6. 61-node Network

Iteration	w/o l.s.	l.s. a-a-o	l.s. o-a-a-t
0	15,041.5	15,041.5	15,041.5
1	163.992	168.815	163.992
2	153.692	162.528	153.574
3	152.620	155.811	152.374
4	152.176	155.212	151.938
5	151.819	154.544	151.571
6	151.564	154.371	151.232
7	151.312	154.053	150.944
8	151.081	153.958	150.751
9	150.886	153.697	150.576
10	150.737	153.672	150.421
11	150.609	153.467	150.291
12	150.491	153.408	150.198
13	150.397	153.273	150.089
14	150.323	153.203	150.018
15	150.258	153.105	149.956

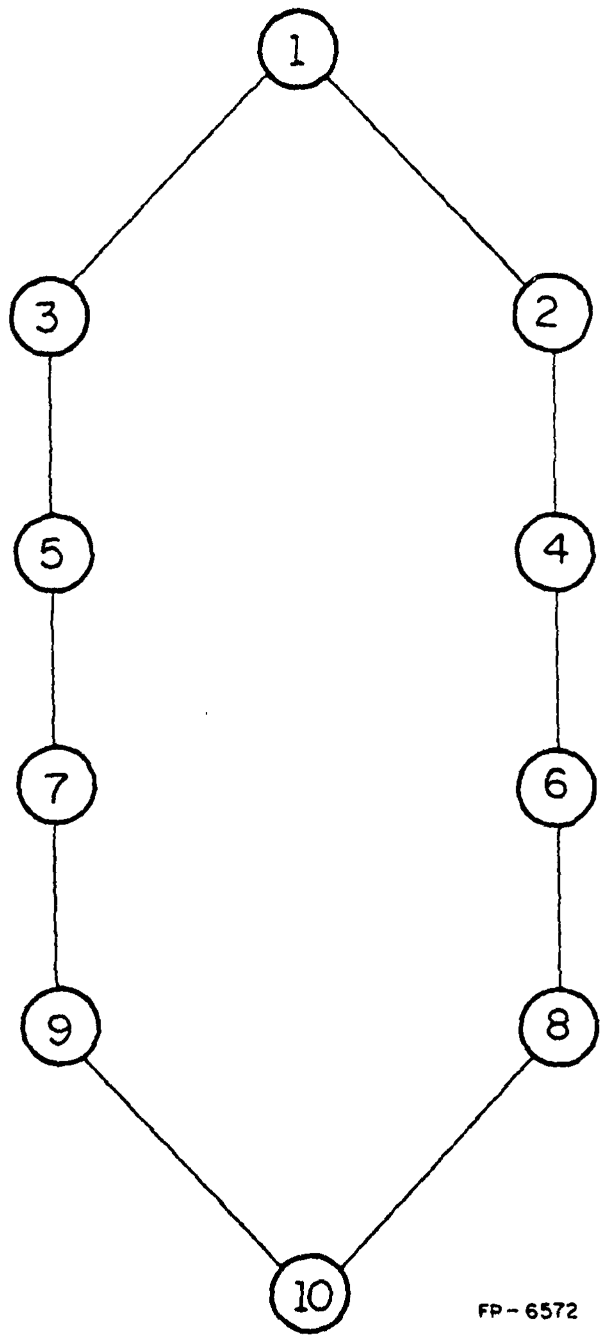
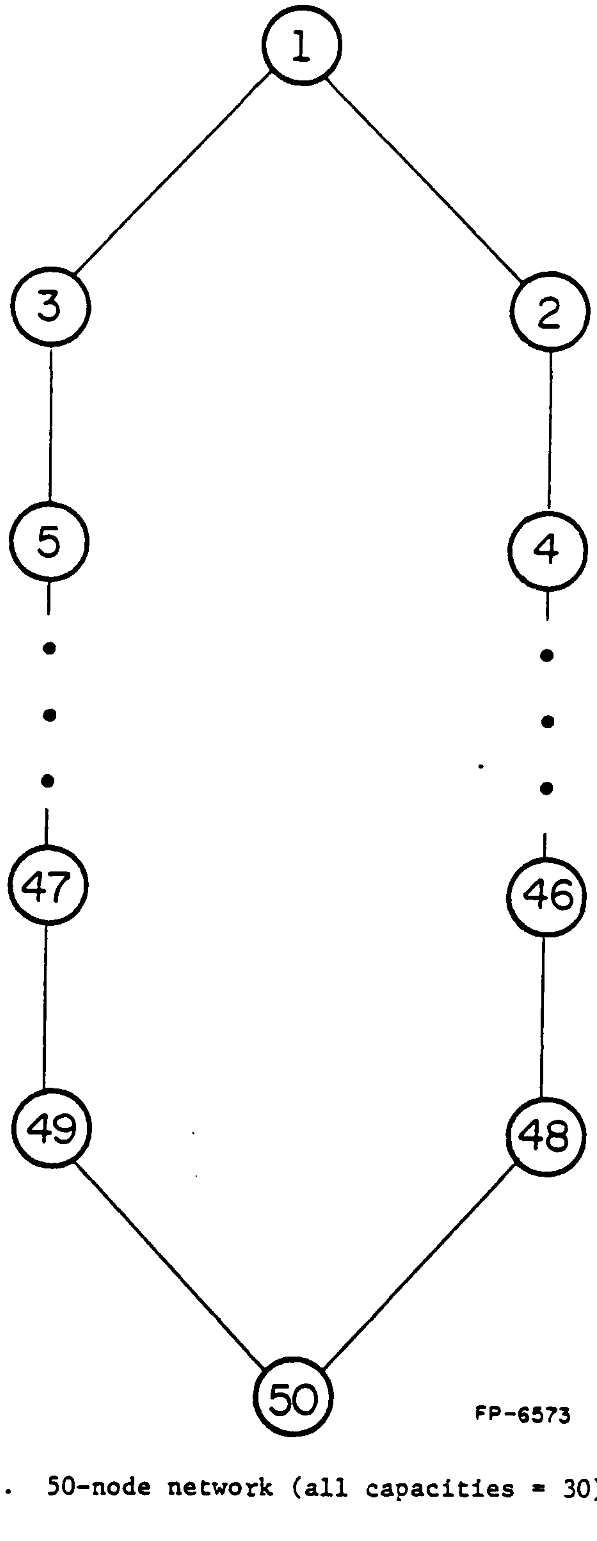


Figure 9. 10-node network (all capacities = 30).



FP-6573

Figure 10. 50-node network (all capacities = 30).

Table 7. 10- and 50-node Networks (all-at-once line search algorithm only)

Iteration	10-node network	50-node network
0	303.484	323.484
1	70.2143	90.2536
2	12.0000	51.9999
3	10.2502	34.5370
4	10.2500	34.3612

CHAPTER 4

MAXIMUM LINK UTILIZATION AS A MEASURE OF DELAY

4.1. The Algorithm

Consider the following multicommodity flow problem [16]

$$\text{(a)}\quad \text{minimize} \left\{ \max_{(i,k)\in\mathcal{L}} \left[\frac{\sum_{j=1}^{N} f_{ik}(j)}{C_{ik}} \right] \right\}$$

subject to

$$\text{(b)}\quad \sum_{k\in O(i)} f_{ik}(j) - \sum_{m\in I(i)} f_{mi}(j) = r_i(j) \qquad \forall i\neq j,\ i\in\mathcal{N},\ j\in\mathcal{N}$$

$$\text{(c)}\quad f_{ik}(j) \geq 0 \quad \forall (i,k)\in\mathcal{L},\ j\in\mathcal{N} \qquad (31)$$

$$\text{(d)}\quad f_{jk}(j) = 0 \quad \forall (j,k)\in\mathcal{L}$$

$$\text{(e)}\quad \sum_{j=1}^{n} f_{ik}(j) \leq C_{ik} \qquad \forall (i,k)\in\mathcal{L}.$$

Let f denote the set of flow variables $\{f_{ik}(j)\,|\,(i,k)\in\mathcal{L},\ j\in\mathcal{N}\}$ and let

$$X = \{f\,|\,f \text{ satisfies (1b)-(1d)}\}$$

and

$$f_{ik} = \sum_{j=1}^{n} f_{ik}(j), \qquad \forall (i,k)\in\mathcal{L}.$$

Note that if there is a set of flow variables $f\in X$ satisfying the constraint (31)(e) then this f has objective function value less than or equal to one. Therefore, if f^* solves problem (31) reformulated without the constraint (31)(e) then it, too, has function value less than or equal to one and satisfies the constraint (31)(e). Thus (31)(e) is superfluous

We may now write (31) more succinctly as

$$\text{minimize} \left[\max_{(i,k)\in\mathcal{L}} \frac{f_{ik}}{C_{ik}} \right] \qquad (32)$$

subject to $f\in X$.

The problem (32) may be solved using an approximation algorithm developed by Bertsekas [12] and based on a method of multipliers due to Kort and Bertsekas [13]. In this procedure we utilize the algorithm of Chapter 2 to minimize a sequence of functions which are better behaved and which successively approximate the objective function in (32)

Consider the following problem which is equivalent to (32)

$$\text{minimize} \left\{ \max_{(i,k)\in\mathcal{L}} \left[\frac{f_{ik}}{C_{ik}} - u_{ik}\right] \right\} \qquad (33)$$

subject to

$$f\in X,\ u_{ik} \leq 0 \quad (i,k)\in\mathcal{L}.$$

Using a multiplier method with exponential penalty function $(\frac{1}{\mu})y[\exp(\mu u)-1]$ to eliminate the constraints $u_{ik} \leq 0$ (see Kort and Bertsekas [13], Bertsekas [12]), the following approximation function is obtained

$$p[f,\mu_n,y^n] = \frac{1}{\mu_n} \log\{ \sum_{(i,k)\in\mathcal{L}} y^n_{ik}\exp[\mu_n f_{ik}/C_{ik}]\} \qquad (34)$$

where $\{\mu_n\}$ is an increasing sequence of penalty parameters and the y^n_{ik} are the multipliers updated via

$$y^n_{ik} = \frac{y^{n-1}_{ik}\exp[\mu_{n-1} f^{n-1}_{ik}/C_{ik}]}{\sum_{i,k\in\mathcal{L}} y^{n-1}_{ik}\exp[\mu_{n-1} f^{n-1}_{ik}/C_{ik}]} \qquad (35)$$

where $f^{n-1} = \{f^{n-1}_{ik}(j) \mid (i,k)\in\mathcal{L},\ j\in\mathcal{N}\}$ minimizes $p[f,\mu_{n-1},y^{n-1}]$ over all $f\in X$ and y^o is chosen such that $y^o_{ik} > 0$, $\forall (i,k)\in\mathcal{L}$ and $\sum_{(i,k)\in\mathcal{L}} y^o_{ik} = 1$. (In the subsequent examples let $y^o_{ik} = \frac{1}{L}$ $\forall (i,k)\in\mathcal{L}$.)

The algorithm proceeds as follows:

<u>Iteration 0</u>: Having chosen μ_o and y^o, minimize $p[f,\mu_o,y^o]$ over X to find f^o. (Note that since $\frac{1}{\mu_o}\log(\cdot)$ is an increasing function we may find f^o by minimizing $\sum_{(i,k)\in\mathcal{L}} y^o_{ik}\exp[\mu_o f_{ik}/C_{ik}]$.)

Iteration n: Choose μ_n, and compute y^n from μ_{n-1}, y^{n-1}, and f^{n-1} by

Determine f^n by minimizing $p[f,\mu_n,y^n]$ over X.

Convergence results for this algorithm may be found in [12],[13].

4.2. Computational Results

The "all-at-once" program with line search described in Section 2.3, was implemented with $D_{i\ell} = y_{i\ell}\exp[\mu f_{i\ell}/C_{i\ell}]$ for each $(i,\ell)\in\mathcal{L}$ where $y_{i\ell}$ and μ were treated as constants. The procedure is terminated when the following condition is satisfied

$$-\sum_{(i,\ell)\in\mathcal{L}} D'_{i\ell}(f^k_{i\ell})[\bar{f}^k_{i\ell}-f^k_{i\ell}] \leq \frac{\varepsilon}{\mu}$$

where $\varepsilon > 0$ is chosen experimentally. (We found ε=.01 satisfactory.) The *multipliers* y_{ik} and the penalty parameter are, then, updated as per Section 4.1: the y_{ik}'s via (35) and μ via $\mu_n = 2\mu_{n-1}$. Finally, the procedure is restarted using the final φ's of the previous overall iteration as the initial φ's for this overall iteration. This is repeated, usually until μ_n becomes too large to exponentiate. An alternative technique could have been implemented to deal with this situation, however in most cases results were satisfactory by the time this problem occurred.

The computational results using this algorithm on the networks of Chapter 3 are given in Tables 8-14. The topologies and inputs are given in Chapter 3. The 11-node network is the only one not tabulated here since the network was designed to exhibit a peculiarity of the algorithms of Chapter 3 and had rather uninteresting results for the algorithm of this chapter.

Table 8. 8-node Network

Iteration	μ	objective function value	dual function value	# of subiterations
1	1	.839651	.571731	3
2	2	.835646	.631242	2
3	4	.821258	.713947	1
4	8	.804470	.782249	2
5	16	.800429	.799000	2
6	32	.800173	.799993	2
7	64	.800289	.800003	1
			total	13

Table 9. 16-node Network

Iteration	μ	objective function value	dual function value	# of subiteration
1	1	1.00000	.421005	1
2	2	.791925	.562110	3
3	4	.793172	.660648	1
4	8	.756496	.729195	2
5	16	.750364	.748519	2
6	32	.750836	.749989	1
7	64	.753084	.750123	2
			total	12

Table 10. 26-node Network

Iteration	μ	objective function value	dual function value	# of subiterations
1	1	.807963	.484548	1
2	2	.790036	.523222	3
3	4	.770317	.576217	3
4	8	.747149	.652613	3
5	16	.744158	.714161	2
6	32	.735663	.731803	3
7	64	.733812	.733302	3
			total	18

Table 11. 32-node Network

Iteration	μ	objective function value	dual function value	# of subiterations
1	1	.811112	.343290	2
2	2	.618275	.419589	3
3	4	.548648	.492498	2
4	8	.537304	.526965	2
5	16	.536741	.533052	3
6	32	.535678	.533344	2
7	64	.535338	.533341	3
8	128	.534271	.533327	3
			total	20

Table 12. 61-node Network

Iteration	u	objective function value	dual function value	# of subiterations
1	1	1.07951	.403508	1
2	2	.941108	.478478	3
3	4	.915899	.624600	2
4	8	.879828	.792732	4
5	16	.866770	.852357	4
6	32	.867209	.863138	19
7	64	.867782	.862657	5
			total	38

Table 13. 10-node Network

Iteration	μ	objective function value	dual function value	# of subiterations
1	1	.897542	.247836	1
2	2	.754853	.501543	1
3	4	.669948	.639721	1
4	8	.666685	.666631	2
5	16	.666667	.666667	2
			total	7

Table 14. 50-node Network

Iteration	μ	objective function value	dual function value	# of subiterations
1	1	.981398	.191372	1
2	2	.952438	.256591	1
3	4	.822027	.450718	2
4	8	.679340	.648845	3
5	16	.666688	.666618	2
6	32	.666667	.666667	1
			total	10

We note that while the dual function value will always increase, there is no guarantee that the objective function will always decrease. In fact in the 8-, 16-, and 61-node networks there is a slight increase in the last few iterations. Of course, we still can be sure that the true optimal objective function value is between the smallest objective function value achieved and the last dual function value.

CHAPTER 5

CONCLUSIONS

The examples of the two preceding chapters are, we feel, representative of our overall computational experience. All of the networks considered, and a number of others, were run with a variety of inputs and the results were generally similar. Two of the networks exhibited (the 26- and 61-node networks) came from early versions of the ARPANET. The others were designed with certain emphasized characteristics. For example, the 16-node network (see Figure 5) symbolizes two presumably distant centers, with "local" traffic, connected by a small number of higher capacity links (e.g. the East and West Coasts of the U.S.A, Europe and the U.S.A.).

The line search procedure used here is complex and consumes more computer time than seems affordable by its success. The examples of Chapter 3 give little evidence of any significant improvement due to use of this line search. The excessive cost is particularly evident in the one-at-a-time where the line search must be performed N times. It is possible that some ad hoc line search might be used in extreme cases (e.g. when the objective function value is not decreased by some percentage).

One problem our particular line search was designed to alleviate is the persistance of improper links. This it did quite well, although with less ease when the utilization was high for several links. However, the existence of improper links in the algorithm without line search seemed to be no impediment to convergence.

In the algorithm of Chapter 4 convergence was relatively swift even when the network was unusually congested (the ARPANET is known to rarely run with average link utilization over 50%).

Since the two measures of delay discussed here are intrinsically different a choice of one over the other would most likely depend on the properties of the network under consideration. On the other hand, given that one had performed the optimization for one measure of delay, one might like to know whether largely different results would have been obtained had one used the other criterion. Table 15 contains the maximum utilization and Kleinrock delay for flows near the optimum in the algorithms of Chapters 3 and 4. With the exception of the 50-node network the figures are quite close. However, the 50-node network is indicative of a situation which can be pushed to the extreme.

Consider a network with N nodes (where for simplicity N is assumed to be even) arranged in an oblong fashion with each node linked to the next (as in the 50-node network) so that the links form an elongated ellipse. Also let the end nodes be 1 and N, and let one of the two nodes linked to 1 be 3. Let the capacity for each link be 30 and let $r_3(1) = r_N(1) = 30$ and $r_i(j) = 0$ for all other (i,j). It is easily seen that the optimal flow for the maximum utilization case is to send all flow from 3 directly to 1 and all the flow from N to 1 so that it does not pass through 3 thereby achieving the minimum of 2/3 maximum link utilization regardless of how large N is. On the other hand, it is a messy but straightforward calculation to show that as $N \to \infty$ the minimum Kleinrock delay becomes infinite.

The Kleinrock delay function for an individual link may be written as $\frac{(f/c)}{1-(f/c)}$. If the sum of such delays is bounded in a collection of network routings then the corresponding maximum utilization is similarly bounded

Table 15. A comparison of the two delays near their respective optima

	Maximum Utilization		Kleinrock Delay	
Network	Chapter 3 Algorithm	Chapter 4 Algorithm	Chapter 3 Algorithm	Chapter 4 Algorithm
8-node	.821	.800	41.9	42.0
10-node	.777	.667	10.3	12.0
16-node	.761	.750	69.1	70.3
26-node	.753	.734	65.3	66.7
32-node	.615	.534	94.8	95.3
50-node	.862	.667	34.4	52.0
61-node	.870	.864	149.8	153.9

away from one. In particular, if $M \geq \frac{(f/c)}{1-(f/c)}$ then $1 > \frac{M}{M+1} \geq \frac{f}{c}$. As shown above the reverse is not necessarily true, however if the number of nodes is also bounded, say by N, then $1 > M \geq \max_{i=1,\ldots,L} \frac{f_i}{c_i}$ implies

$$\infty > \frac{NM}{1-M} \geq \sum_{i=1}^{L} \left(\frac{f_i}{c_i - f_i}\right).$$

REFERENCES

1. D. P. Bertsekas, "Algorithms for Optimal Routing of Flow in Networks," Coordinated Science Laboratory Working Paper, University of Illinois at Urbana-Champaign, revised June 1978.

2. R. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE Trans. on Communication, Vol. COM-25, pp. 73-85, 1974.

3. E. S. Levitin and B. T. Polyak, "Constrained Minimization Problems," USSR Comput. Math. Math. Phys., Vol. 6, pp. 1-50, 1966.

4. A. A. Goldstein, "Convex Programming in Hilbert Space," Bull. Amer. Math. Soc., Vol. 70, pp. 709-710, 1964.

5. P. Halmos, Naive Set Theory, D. Van Nostrand, Princeton, N.J., 1960.

6. S. E. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms," Operations Research, Vol. 17, pp. 395-412, 1969.

7. R. W. Floyd, "Algorith 97, Shortest Path," Comm. ACM, Vol. 5, p. 345, 1962.

8. S. Warshall, "A Theorem on Boolean Matrices," J. ACM, Vol. 9, pp. 11-12, 1962.

9. N. W. Smith, SAIL Tutorial, Memo AIM-290, Stanford Artificial Intelligence Laboratory, Oct. 1976.

10. John Reiser (ed)., SAIL, Memo AIM-289, Stanford Artificial Intelliegence Laboratory, Aug. 1976.

11. D. G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Mass., 1973.

12. D. P. Bertsekas, "Approximation Procedures Based on the Method of Multipliers," J. Opt. Theory and Applications, Vol. 23, pp. 487-510, 1977.

13. B. W. Kort and D. P. Bertsekas, "A New Penalty Function Method for Constrained Minimization," Proc. of 1972 IEEE Conf. on Decision and Control, New Orleans, La., pp. 162-166, 1972.

14. E. M. Gafni, "Convergence of a Routing Algorithm," M.S. thesis, Dept. of Electrical Engineering, University of Illinois, Urbana, 1979.

15. L. Kleinrock, Queuing Systems, Vol. I, Wiley, New York, 1976.

16. J. Defenderfer, "Comparative Analysis of Routing Algorithms for Computer Networks", Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass, 1977, (also Electronic Systems Lab., report no. 756, M.I.T.).